Understanding Protein Mechanism With Computational Approaches

by

Yunlong Liu

A dissertation submitted to The Johns Hopkins University

in conformity with the requirements for the degree of

Doctor of Philosophy

Baltimore, Maryland

May, 2018

Abstract

Proteins, which are essentially linear polymers of amino acid residues, perform a tremendous variety of functions, including catalyzing biochemical reactions, replicating genetic information, transporting molecules, constructing cell architecture, and transducing signals in the microscopic milieu of cells and the interstitial fluid. Such linear chains of residues undergo complicated folding processes to form biologically active 3-dimensional structures as they are translated. Since the first protein structure of myoglobin was determined using X-ray crystallography in 1958, biophysicists have been making continuous efforts to understand the associations between protein structures and their functions. Unfortunately, only static snapshots and limited dynamics of proteins at atomic-level resolution are available with the experimental approaches developed so far, which precludes a complete biophysical and temporal understanding of protein mechanisms in the context of the microscopic biochemical processes in which they operate. Molecular dynamics (MD) simulations have been introduced as a computational approach to acquire information about the full dynamics of molecular systems, especially proteins. The method allows for sampling of the time evolution of proteins in the form of trajectories (a collection of frames containing precise coordinates), provides quantitative insights into how proteins function, and has been widely used as an essential tool to understand protein mechanisms.

Specialized computer hardware developed in recent years, coupled with improved software implementations, have greatly accelerated MD simulations. A single GPU-attached cluster is capable of producing microsecond-length trajectories in reasonable amounts of time. As the timescales of MD simulations keep increasing, it becomes more challenging to analyze long trajectories. Tasks such as identifying the conformational

states of proteins by clustering frames in trajectories are usually resolved with unsupervised machine learning algorithms. The quality of their results directly affects the answers of many biophysical questions, which implies that using an appropriate analytical method for a particular MD aim is as important as a sufficient sampling of protein conformations. In the first two chapters of this dissertation, the basic theory of MD simulations is introduced together with several important unsupervised learning algorithms we used in investigating specific biophysical problems, which target the issues of dimensionality reduction and clustering. These algorithms include the novel data-driven and generative clustering algorithm we developed based on adversarial autoencoders. Then we present our investigation on the conformational dynamics and ion transport mechanism of the sodium iodide symporter, which could be considered as an example of adopting MD and unsupervised learning approaches to understand the working mechanisms of proteins. In the last chapter, we introduce an application of computational structural biology to large-scale drug screening.

Thesis Committee

Primary Readers

L. Mario Amzel (Primary Advisor)
Professor
Department of Biophysics and Biophysical Chemistry
Johns Hopkins University School of Medicine

Thomas B. Woolf (Thesis Committee Chair)
Professor
Department of Physiology
Johns Hopkins University School of Medicine

Other Members

Sandra Beatriz Gabelli
Associate Professor
Department of Biophysics and Biophysical Chemistry
Johns Hopkins University School of Medicine

Albert Y. Lau
Assistant Professor
Department of Biophysics and Biophysical Chemistry
Johns Hopkins University School of Medicine

Acknowledgments

With my greatest sincerity, I would like to thank my advisor Prof. L. Mario Amzel first, for his support of my Ph.D. research and his critical reading of this dissertation, for his patience, for his open mind and for his kindness. If it wasn't for his profound knowledge and open mind, my research couldn't have gotten this far. If it wasn't for his patience and tolerance, my Ph.D. life wouldn't be this happy. The experience of studying and working with him will always shine in my memory.

Prof. Tom Woolf, Prof. Sandra Gabelli, and Prof. Albert Lau, who comprise my thesis committee, have provided me substantial guidance in my research as well as in my career. None of them have missed any of the thesis meetings or discussions in the past 5 years. I thank the committee for all of their kindness, patience and assistance. Particularly, I thank Prof. Tom Woolf for volunteering to be the reader of my thesis. I will remember all of these nice and passionate academic thinkers forever.

I thank the Biochemistry, Cellular, and Molecular Biology (BCMB) Graduate Program for providing me the opportunity to pursue my scientific interests, and for giving me a foundation for my future career development.

I thank my mother, Ping Han and my father, Jun Liu for giving me life, for spending a very large proportion of their limited income in my education, for their unconditional support and hearing, for their consistent efforts on making me happy.

Special thanks go to my wife, Yijie Li, whom I met in the second year of my Ph.D. and with whom I constituted a family in the last year. She is my best companion, closest friend and life-long consultant, who shares all the pressure and sorrow of life and all the happiness that life offers. I wouldn't complete my Ph.D journey without her lovely support and consolation.

I acknowledge my scientific collaborators, Prof. Nancy Carrasco, Prof. Gregg Semenza, and Dr. Ignacia Echeverria, who shared with me their passion for scientific research and who helped me with my science. I also thank the Maryland Advanced Research Computing Center (MARCC) and its staff for generously providing me precious computational facilities and prompt technical support as well as D.E. Shaw Research for kindly offering us their trajectory.

Last but not least, I would like to thank my dear friends in the Amzel lab: Mayukh Chakrabarti, Jesse Yoder, and Dr. Harry Saavedra, for all our discussions on politics, culture and science. Especially, I thank Mayukh, the other computational hacker in the lab, for sharing the room with me and for his valuable suggestions about my writing.

Table of Contents

Table of Contents vii

List of Tables x

List of Figures xi

1 Introduction 1

1.1 Protein Dynamics 1

1.2 MD Simulations 5

1.3 Analytical Approaches 8

1.4 Outline 10

1.5 Structure of Fiesta3 10

1.6 MARCC Computational Facility 12

2 Machine Learning Models: Theory and Applications in Analyzing MD Simulations 15

2.1 Notations 15

2.2 Dimensionality Reduction 15

2.2.1 Linear PCA 16

2.2.2 Kernel PCA 18

2.2.3 Deep Autoencoder 19

2.3 Clustering MD Trajectories 23

2.3.1 Introduction 23

2.3.2 Theory 25

2.3.2.1 KMeans and KMedoids 25

2.3.2.2 Generative Adversarial Network 27
2.3.2.3 Adversarial Autoencoder (AAE) 28
2.3.2.4 Reparameterization Trick 31
2.3.2.5 AAE with Gumbel-Softmax 36
2.3.3 Experiments and Results 39
2.3.3.1 A Testing Model for Clustering Algorithms: Trp-Cage . 39
2.3.3.2 Clustering with KMedoids 40
2.3.3.3 Clustering with AAE 43
2.3.3.4 Clustering with AAE-GS 45
2.3.3.5 Generation of fake frames 45
2.3.4 An application of clustering frames with AAE-GS 48
2.3.5 Discussion and Conclusion 53

3 Ion Transport Mechanism of NIS 59

3.1 Introduction 59
3.2 Methods 61
3.2.1 MD Simulations 61
3.2.1.1 The Construction of NIS Simulation System 61
3.2.1.2 Energy Minimization and Equilibration 63
3.2.1.3 Non-Restraint Production Simulations 65
3.2.1.4 Restraint Simulations 66
3.2.2 AAE-GS Clustering 68
3.3 Results 70
3.3.1 Restraint-Induced Conformation Change 70
3.3.2 Validation of Ion Binding Sites 72
3.3.3 Transport of NaA and the Iodide Ion 75
3.4 Conclusions 75

4 Large-Scale Structure-Based Virtual Screening 90

4.1 AutoDock Vina 91

4.2 SparkVina 92
4.2.1 Python Interface for Vina 93
4.2.2 Data Processing Workflow 95
4.2.3 Running Configurations 95
4.3 Experiments 97
4.3.1 SBVS Experiment 97
4.3.2 Scalability 98
4.4 Future Work 99

5 Discussion and Conclusions **104**

Appendix A Understanding the mechanism of physiological effectors and oncogenic mutations in the activation of PI3K: A computational approach **106**

A.1 Introduction 106
A.2 Methods 108
A.2.1 The Construction of MD Simulation System 108
A.2.2 MD Simulations 109
A.2.3 Data Analysis 109
A.3 Preliminary Results[1] 110
A.3.1 Fluctuations Enhanced by the Mutations 110
A.3.2 Covariance between Different Domains 110
A.3.3 Mode of Motions 111

[1]Since this research is not finalized, only a few preliminary results are presented in this section.

List of Tables

2.1 Clarifications and explanations on symbols and notations 16

2.2 Structures of all sub-components of our AAE model. 43

3.1 Other parameters used in equilibration runs. k in unit kJ/nm^2 and lengths in unit picosecond (ps). "dt" represents the step size, which is in unit femtosecond (fs). 65

3.2 NIS trajectories. k_{pr} in unit kJ/nm^2 ($kcal/\text{Å}^2$) 67

3.3 Structures of all sub-components of our AAE-GS model. 69

3.4 A distribution of frames grouped by their cluster ID and the trajectories they are sampled from. 73

List of Figures

1.1 Chapter Dependencies 11
1.2 Simplified, High-Level Structure of Fiesta3 12

2.1 A simple example of a feed-forward neural network with one hidden layer 20
2.2 The general architecture of AAE for clustering. 30
2.3 A general architecture of AAE with Gumbel-softmax. The architecture is constructed similarly as AAE. However, $p(\alpha)$ illustrates a Gumbel(0, I) distribution, which combines the variable $p(\pi)$ to form the categorical representation. 38
2.4 Visualization of cluster representatives obtained from KMedoids experiment. Ten frames randomly selected from each cluster are shown as the representatives in each row. All structures are oriented to a fixed orientation. The estimated probability mass for each cluster is listed in brackets underneath each cluster id label. 41
2.5 Convergence of AAE and AAE-GS clustering experiments. The restruction losses plotted in A of both experiments were evaluated every 2000 steps. B and C plots the first two dimensions of the "learnt" style vectors of all frames in AAE and AAE-GS experiments, respectively. 42
2.6 Visualization of cluster representatives obtained from AAE experiment. The first ten frames of each cluster sorted by the norm of the style vector are shown in each row. All structures are oriented to a fixed orientation. The estimated probability mass for each cluster is listed in brackets underneath each cluster id label. 44

2.7 The annealing process of the parameter τ. 46
2.8 Visualization of cluster representatives obtained from AAE-GS experiment. 47
2.9 Examples of fake frames generated from the AAE-GS model. The corresponding Gaussian noise vectors are plotted as bar plots. All 16 elements of the style representation are spread out along the horizontal axis of each bar plot. Different dimensions are plotted in different colors. 49
2.10 Structural descriptions for selective states. The backbone of the protein is plotted with the representation Cartoon and key residues are shown with balls and sticks. 51
2.11 Transitions between different states of Trp-cage. The first frame in each cluster is used as the representation of the cluster. The unfolded state is circled in blue. The perfect folded state is double circled in black. A misfolded state (State 1) is circled in red dashes. All transition probabilities are sit in the middle of the arrow. Arrows with a transition probability lower than 0.005 are pruned. 52
2.12 Random frames selected from the first 10,000 of each cluster (AAE) . . 54
2.13 Random frames selected from the first 10,000 of each cluster (AAE-GS) 55

3.1 Previously proposed working mechanism of NIS. A–H represents different conformational states of NIS, which favor different ion binding behaviors. This mechanism assumes the order of ion bindings and ion releases. No ion translocation inside the tunnel of the transporter is assumed. [13] . 62
3.2 NIS simulation system. The protein in the middle of the system is plotted in the cartoon representation. POPC lipid bilayer is shown in purple points and DLPC lipid layer in orange points. All water molecules are presented in red points. Ions are generally plotted with the hyperball representation. Na^+, K^+, Cl^-, I^- ions are colored with yellow, orange, gray and white, respectively. 64

3.3 Convergence of AAE-GS. The convergence of the autoencoder loss is plotted in A and the first two dimensions of the trained style representations are shown in B. 78

3.4 The validation of the outward-open conformation from the extracellular perspective. Figure A, B and C represent protein in Cartoon, where SaiT is colored in blue, C6 colored in red and C15 colored in green. A degree of opacity is used for plotting SaiT to avoid unclear overlap between structures. Figure D, E and F plots the electrostatic surfaces of C6, C15 and SaiT, respectively. Arrows with a same color are used to mark a local region subject to structural comparision. 79

3.5 The validation of the inward-open conformation from the intracellular perspective. Figure A, B and C represent protein in Cartoon, where SaiT is colored in blue, C6 colored in red and C15 colored in green. A degree of opacity is used for plotting SaiT to avoid unclear overlap between structures. Figure D, E and F plots the electrostatic surfaces of C6, C15 and SaiT, respectively. Arrows with a same color are used to mark a local region subject to structural comparision. 80

3.6 Frequent contacts between I^- and residues. Frame indices are plotted on the vertical axis and residues were displayed on the horizontal axis. Residues making accumulated contacts shorter than 2 ns are abandoned. 81

3.7 Frequent contacts between NaA and residues. Frame indices are plotted on the vertical axis and residues were displayed on the horizontal axis. Residues making accumulated contacts shorter than 2 ns are abandoned. 82

3.8 Frequent contacts between NaB and residues. Frame indices are plotted on the vertical axis and residues were displayed on the horizontal axis. Residues making accumulated contacts shorter than 2 ns are abandoned. 83

3.9 Na2 site. NIS is represented in Cartoon and key residues composing Na2 Site are represented as balls and sticks. 84

3.10 I^- binding pockets. NIS is represented in Cartoon and key residues composing the I^- site are represented as balls and sticks. The I^- is represented as the white hyperball. Figure A and B provides side views of the protein as well as those key residues. Figure C and D represent the perspective from the extracellular side to the intracellular side 85

3.11 NaA binding pockets. NIS is represented in Cartoon and key residues composing the binding pockets are represented as balls and sticks. The Na1 is represented as the yellow hyperball. Figure A and B provides side views of the protein as well as those key residues. Figure C and D represent the perspective from the extracellular side to the intracellular side 86

3.12 Side chain orientations of M258. Figure A plots the extracellular side pointing orientation of M258 side chain when I^- binds the upper pocket. Figure B shows the M258 side chain becomes flat after NaA moves throught the upper pocket. The protein is shown in Cartoon. Na^+, I^- are shown in yellow and white hyperballs, respectively. M258 are represented with balls and sticks. The protein is clipped near to avoid blocking M258. . . 87

4.1 RDD (Resilient Distributed Datasets) transformation flow implementing SBVS 96

4.2 Compounds docked to the HIF-2α. 98

4.3 The binding conformation of the "best" compound we identified 99

4.4 Running Time vs. Number of CPUs 100

4.5 Speedup vs. Number of CPUs 100

A.1 Structure of the different domains of p110α in complex with niSH2 domains of p85α. This representation is made based on the crystal structure (PDB ID: 3HIZ) 112

A.2 RMSF and Protein Fluctuations. Three columns of this figure, from left to right, represent the protein fluctuations of WT, R38CR88Q, R38CR88QN345K, respectively. Each column is labeled as a pair, e.g., A and A'. Figure A, B and C give the RMSF per residue, and A', B' and C' reflect the RMSF value on the structure of the protein, which is colored by a color gradient from blue (least fluctuated) to red (most fluctuated). 113

A.3 Covariance Matrices. The covariance matrices of WT, R38CR88Q and R38CR88QN345K are represented by two heatmaps. The elevated signals of the covariance in the mutants are highlighted by the red rectangles. . 114

A.4 Visualization of the first two modes of motion idenitified by PCA. Green arrows indicate moving directions of residues in each mode. The length of the arrows represents the amplitude of the motion. 115

Chapter 1

Introduction

In recent years, with the advent of single-particle electron cryo-microscopy (Cryo-EM) and nuclear magnetic resonance (NMR), the conformational dynamics of proteins can now be visualized at near-atomic resolution [1]. However, understanding protein mechanisms at atomic resolution still remains a challenge in the biophysics community. As a complement to experimental methods, computational approaches can provide insights into protein mechanisms by sampling protein dynamics from a set of protein coordinates at a certain time scale and quantitatively analyzing the conformations sampled. In this dissertation, we present existing computational approaches in combination with a novel method that we have developed for analyzing protein conformations. We then demonstrate the application of these methods through investigations on the mechanisms of two protein systems. The remainder of this chapter provides a brief overview on the logic of the computational approaches used in our investigations, presents an outline of the following chapters, and summarizes the code architecture associated with our implementation.

1.1 Protein Dynamics

Proteins, serving as building blocks of living organisms, are essentially linear polymers of amino acid residues which can be represented by their sequences. In order to perform its biological function, the chain of amino acid residues covalently linked by peptide bonds folds into a native 3-dimensional structure uniquely determined by the sequence of the peptide. Moreover, a variety of biophysical experiments showed that rather than

being static structures, proteins are dynamic systems that move through an ensemble of conformations at room temperature [2]. These conformations can be grouped into several globally distinct macrostates which are associated with their observed behaviors and the conformations that are grouped into each macrostate based on their similarity are named as "conformational substates" [2]. Strictly speaking, proteins are objects that live in a non-degenerative high dimensional topological space X. Assuming that the protein system has N degrees of freedom, the time evolution of the protein system can generally be described by the time-dependent Schrödinger equation $i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle$. Protein systems can be considered special cases of many-body systems. $\hat{H}$, the Hamiltonian, can be represented as

$$\hat{H} = \sum_{i}^{N} -\frac{\hbar^2}{2m_i}\nabla_i^2 + U(\mathbf{r}_1, \ldots, \mathbf{r}_N) + V(t) \tag{1.1}$$

where m_i is the mass of the i-th particle and U is the potential operator. An energy perturbation term $V(t)$ is added to the Hamiltonian to account for the fact that the protein system is not an isolated system and is observed to have constant energy exchange with the environment. For instance, some channel proteins react to electrochemical potentials across the lipid bilayer. Technically, involving $V(t)$ makes $\hat{H}$ time-dependent.

A simple, semi-quantitative analysis of the Schrödinger equation of such a many-body system with energy perturbation provides a broad view on protein dynamics. To start with, we temporarily omit the effect of $V(t)$. Then a solution $|\psi(t)\rangle$ of the unperturbed time evolution equation can be written as a linear combination of energy eigenstates $|i\rangle \mid i \in N$ (Shown in Equation 1.2), where each eigenstate is corresponding to an eigenvalue of the Hamiltonian.

$$|\psi(t)\rangle = \sum_{j} c_j(t)\,|j\rangle \tag{1.2}$$

where $c_j(t) = c_j(0)e^{-iE_jt/\hbar}$. Any quantum measurement of the system will cause it collapses to an eigenstate $|i\rangle$ with a probability of $|c_i|^2$ assuming that $\psi(t)$ is normalized. This shows that the uncertainty of the energy state of the system exists intrinsically and so does the energy fluctuation.

If $V(t)$ is taken into consideration and we assume it is deterministic, a usual way to solve the time-dependent Schrödinger equation is to assume a solution following the form of $\psi(t) = \sum_k c_k(t) \left|k\right\rangle e^{-iE_k t/\hbar}$. After substituting this form into the Schrödinger equation, it becomes

$$i\hbar \sum_k \frac{\partial c_k(t)}{\partial t} e^{-iE_k t/\hbar} \left|k\right\rangle = \sum_k V(t) c_k(t) e^{-iE_k t/\hbar} \left|k\right\rangle \tag{1.3}$$

For every n, we apply $\left\langle n\right|$ to both side of Equation

$$\frac{\partial c_n(t)}{\partial t} = \frac{1}{i\hbar} \sum_k V_{nk}(t) c_k(t) e^{-i(E_n - E_k)t/\hbar} \tag{1.4}$$

Equation 1.4 indicates that in responding to the external energy perturbation, coefficents of each eigenstates are connected during time evolution of the protein system. As mentioned above, since the coefficients $c_k(t)$ represent the probability of each eigenstate, transition probabilities can be computed based on Equation 1.4. This illustrates that constant energy exchange has a great impact on the quantum state transition of the protein system.

In reality, Hamiltonians for real protein systems are so complicated that quantum mechanics cannot give analytical descriptions of protein dynamics. The numerical integration of the time evolution equations is also computationally infeasible. In favor of the need to explain experimental observations and to make empirical predictions, the free-energy landscape model [2] [3] [4] was proposed and is the most used by the biophysics community. The model is based on the fact that the potential energy of the protein system is merely determined by its conformation. In other words, there exists a function that maps any point in X to an energy dimension $\mathbb{R}$. This function depicts a manifold ("hypersurface" or "energy landscape") in $n+1$ dimensional space, which very likely contains many local minima and maxima. This model is in agreement with the thermodynamic hypothesis of protein folding [5] in which the protein naturally folds to is a conformation corresponding to a local minimum on the energy landscape, the native structure. And constant energy exchange can drive state transitions of the protein systems if the barriers between local minima are smaller than the energy fluctuations.

This hypothesis is controversial [6] and it turns out that at some cases, conformations with local minimized energy are not kinetically accessible. Nevertheless, protein systems can definitely explore more subsets of X with constant energy exchange no matter where their native structures locate in the conformational space. In addition, binding to ligands alters the energy landscape of the protein system in favor of its state transition.

Concluding from the above analysis, protein dynamics is generally caused by the intrinsic quantum state uncertainty of the protein system, constant energy exchange between the protein systems and the environment, or ligand binding behaviors. Usually, a composition of these events results in a working mechanism for a protein system. Due to this close linkage between the dynamics and the working mechanism of the protein, having a sufficient sampling of protein dynamics is crucial for revealing how it functions.

Unfortunately, current experimental approaches are unable to capture the dynamics of protein at atomic resolution. X-Ray Crystallography is used to determine high resolution structures through packing proteins into crystal cells, which locks proteins into one of their stable conformations. It is known that resolving the structures of flexible regions of the protein or of natually disordered proteins is generally hard for this approach. Cryo-EM, which is a recently developing technique, determines protein conformations at near-atomic resolution, but it is impossible to estimate transition rates between all the conformations since it lacks temporal continuity. Nuclear Magnetic Resonance (NMR) becomes vulnerable when the size of the protein in question gets larger, and it requires proteins to be steady in solution for hours or even days. Other emerging experimental techniques, such as Time-Resolved X-Ray Crystallography, still have numerous unresolved challenges, which make them generally unapplicable.

In contrast, molecular dynamics (MD) simulations are computational methods uniquely suited to sample protein conformations and quantitatively analyze conformational changes. The collection of conformations they sampled are in the form of temporally ordered trajectories, which makes quantifying state transitions possible. To sufficiently sample protein dynamics, MD simulations are usually performed for significant time spans.

Dozens of analytical methods, including statistical methods and machine learning algorithms, are thus required for organizing and extracting information from the produced long trajectories. It is proven by extensive research that MD simulations combined with feasible analytical methods can provide important physical insights which are not inferrable from experimental observations. In this dissertation, we adopt these computational approaches to sample protein dynamics and study the working mechanisms of proteins.

1.2 MD Simulations

An MD simulation refers to a computer simulation that attempts to describe the time evolution of a particular molecular system by integrating differential equations. Such molecular system is usually constructed by sitting macromolecules in the center of a finite size space-filling box and then either wrapping them with solvent molecules (explicit-solvent) or embedding them in a dielectric region (implicit-solvent). For explicit-solvent simulations, periodic boundary conditions are applied to avoid edge effects caused by putting the systems in vaccum. As introduced in the previous section, a highly accurate description on the time evolution of a system is provided by the time-dependent Schrödinger equations. As integrating those equations numerically at a time scale of interest is impossible for current high performance computing (HPC) facilities, MD simulations fall back to the world of classical mechanics. Rather than using Equation 1.1, they describe the time evolution of the simulation system simply with Newton's second law,

$$m_i \frac{\partial^2 \mathbf{r}_i}{\partial t^2} = -\frac{\partial V}{\partial \mathbf{r}_i}, i = 1 \dots N \tag{1.5}$$

where V represents the potential function and $\mathbf{r}_i$ represents the position of i-th atom in the defined simulation system. The definition of $\mathbf{r}_i$ shows that MD simulations treat every atom as a point in 3-dimensional euclidean space without considering quantum effects of its nucleus and electrons. Thus, they are not able to describe bond formations or electron transfers.

The potential function V is referred as "force field" in the context of molecular

modeling. The definition of V is generally given by

$$\begin{aligned} V &= V_{bonded} + V_{nonbonded} \\ &= V_{bond} + V_{angle} + V_{dihedral} + V_{electrostatic} + V_{vdw} \end{aligned} \tag{1.6}$$

All the terms of V could be written as analytical functions of $\mathbf{r}_i$s with a given set of parameters. These functions come from laws of classical physics. For instance, the nonbonded electrostatic interaction $V_{electrostatic}$ is described using Coulomb's law and the Van der Waals interaction is approximated with Lennard-Jones potential. The parameters of these functions can be defined empirically by fitting to quantum mechanics results and they could also be refined or adjusted according to the practical experience of researchers. Multiple sets of force field has been published and widely used in the computational biophysics community, such as, CHARMM series [7], AMBER99SB-ILDN [8], etc.

After all parameters in Equation 1.5 are defined, MD simulations can be done by integrating it in very small time steps (1 ˜2 femtoseconds). The steps of the naive algorithm are listed below:

1. Initialize the phase configuration. The starting conformation of the protein should be initialized to either an experimentally determined structure or a proper homology model. For explicit-solvent simulations, water molecules and ions are added at random position to defined simulation boxes. The whole system should undergo a robust energy minimization process to avoid structure clashes. The initial velocities should be randomly sampled from the following Maxwell-Boltzmann distribution,

$$p(v_i) = \sqrt{\frac{m_i}{2\pi kT}} \exp\left(-\frac{m_i v_i^2}{2kT}\right), i = 1, \ldots, 3N \tag{1.7}$$

2. Compute forces based on $\mathbf{r}(t)$ and Equation 1.5.

3. Update phase configuration with the Verlet integrator [9].

$$\begin{aligned}
\mathbf{v}(t+\frac{1}{2}\delta t) &= \mathbf{v}(t) + \frac{\delta t}{2m}\mathbf{F}(t) \\
\mathbf{r}(t+\frac{1}{2}\delta t) &= \mathbf{r}(t) + \delta t\mathbf{v}(t+\frac{1}{2}\delta t) \\
\mathbf{v}(t+\delta t) &= \mathbf{v}(t+\frac{1}{2}\delta t) + \frac{\delta t}{2m}\mathbf{F}(t+\delta t)
\end{aligned} \tag{1.8}$$

4. Repeat Step 2 and 3.

In fact, the actual workflow of MD simulations is far more complicated than the above. The complication is caused by reasons of two categories: 1) Some constant macroscopic thermodynamics properties have to be maintained throughout the simulations. 2) The algorithms used in each step of the workflow should be efficient and hardware friendly.

For simulating a normal physiological process, the system should be at least be kept in an environment with constant temperature and pressure. For investigating a macromolecule's behaviors under particular conditions, a temperature gradient may be applied to the simulation as well. It is necessary to properly maintain these macroscopic thermodynamics properties of the system during long simulations because numerical integration errors, force truncation or heating due to external forces may drift those ensemble averages. Coupling algorithms are used to remove slow drifts on those thermodynamic properties. For example, Berendsen temperature coupling algorithm is quite efficient for relaxing systems to a target temperature, whereas Nosé-Hoover temperature coupling is considered as an accurate but empirical algorithm for maintaining the temperature of the equilibrated systems. Similar to temperature coupling, pressure coupling algorithms are applied to gain control over the system's pressure. Incorporated with these algorithms, extra empirical terms are added to Equation 1.5 which makes the update step more complicated.

MD simulations are computationally intensive. At every time step, the potential of the system and its first derivative with respect to each coordinate has to be evaluated to calculate forces. Evaluations of those bonded terms take $O(N)$ time because

the protein is actually a polypeptide chain. However, the nonbonded terms are quite expensive to compute since every pair of atoms that comes close to each other in space needs to be considered. Therefore, evaluating nonbonded terms generally takes $O(N^2)$ time to complete and for explicit-solvent systems, N is usually quite large. Due to this fact, multiple parallelization algorithms such as domain decomposition were developed for launching production MD simulations in high performance computing clusters. Furthermore, existing implementations of MD simulations [10] [11] accelerate the evaluation of nonbonded interactions with multiple GPU cards. Many heuristics and algorithm-level optimizations are applied in those implementations and these improvements made the actual update algorithm far more complicated than what is presented in Equation 1.8. These efforts enable researchers collecting microsecond scale long MD trajectories with affordable hardwares or accessible resources.

1.3 Analytical Approaches

By analyzing MD trajectories, one can obtain a quantitative understanding about protein function, conformational changes and the relationships between the two. This understanding sheds light for designing experiments to reveal the working mechanisms of proteins. Because of the noisy nature and the increasing length of MD trajectories, various analytical approaches need to be integrated to extract useful information. The approaches commonly used in MD research can be roughly categorized into statistical data analysis, dimensionality reduction techniques, clustering methods and probabilistic graphical models.

Calculating statistical measures such as means, variances, correlations and histograms of some quantities of interest is commonly used to describe structural fluctuations and protein-ligand interactions. Direct comparisons of some statistical quantities computed from MD trajectories starting from different configurations can reveal and estimate the effects of those configurations which generally includes mutations, binding of ligands, electrochemical potentials, thermodynamic properties, etc.

As shown in previous sections, protein conformations sampled in MD simulations live

on manifolds (hypersurfaces) in a high dimensional topological space. This is because the motions of protein systems are restrained by force fields. These manifolds, essentially defining the conformational space available to the protein, can be approximated well by lower-dimensional manifolds spanned by another basis set. The process of finding these lower-dimensional manifolds, as well as the new basis set, is referred to as dimensionality reduction. An intuitive benefit of dimensionality reduction is the fact that it eliminates the noise of protein dynamics, e.g, loop fluctuations. A common dimensionality reduction technique widely used in the biophysics community is principal component analysis (PCA). PCA aims to identify principal directions that can maximally explain the variance of protein motions. These directions represent the dynamic modes of the protein systems, which usually have important biophysical relevance. Other dimensionality reduction approaches, for example, independent component analysis (ICA) and its variants (time-dependent ICA and kernel tICA) [12, 13] were also proposed for targeting specific questions.

Due to the fact that the number of frames in sufficently sampled MD trajectories is quite large, clustering algorithms are crucial for identifying macrostates of proteins lying in those trajectories and offering a distribution of those states. Moreover, clustering frames usually serves as the first step of studying transitions between states of the proteins, which implies that it is the key to obtain accurate descriptions of state transitions. Common clustering algorithms used in the community are KMeans, KMedoids, GROMOS[14] and hiearchical clustering. They are generally available in a few MD analytical packages [15, 16]. Besides the existing methods, we propose a novel, data-driven and generative clustering method – adversarial autoencoder with Gumbel-softmax (AAE-GS) to enhance the quality of clustering.

Principal directions, key conformations and independent components are actually common data patterns embedded in our dynamics data. In contrast, probabilistic graphical models such as Markov models (MM) [17, 18] are necessary to provide insights on understanding relationships between data patterns, or more generally, encoding interactions between various type of protein behaviors or logical links between structural

changes and protein functions.

Except the straightforward calculations of statistical measures, other methods mentioned above can be collectively referred as machine learning models. Details of the machine learning models related to this dissertation will be described in the next chapter. The application of analytical methods on practical problems will be introduced in the context of resolving the problems.

1.4 Outline

The outline of the following chapters is:

- Chapter 2 demonstrates computational methods for analyzing MD trajectories, which include simple statistical methods, existing machine learning models and the clustering approach we proposed.
- Chapter 3 shows how to use computational methods to solve the problems posed in Section 1.3.
- Chapter 4 presents an application of computational structural biology on large-scale virtual screening. This chapter doesn't depend on Chapter 2 and 3. Readers interested in large-scale structure-based virtual screening can read this chapter directly.

Figure 1.1 shows the dependency tree of each chapter.

1.5 Structure of Fiesta3

For the convenience of discussing our implementation of analytical methods and distributed job configurations, we illustrate our internal code repository, namely, Fiesta3, here as a reference for the following content.

Fiesta3 is a monolithic source repository that contains code developed for different projects. The purpose of using a single repository is to solve complicated code dependency issues. For example, a same library could be referenced in the code bases of different projects. To facilitate building code written in different programming languages

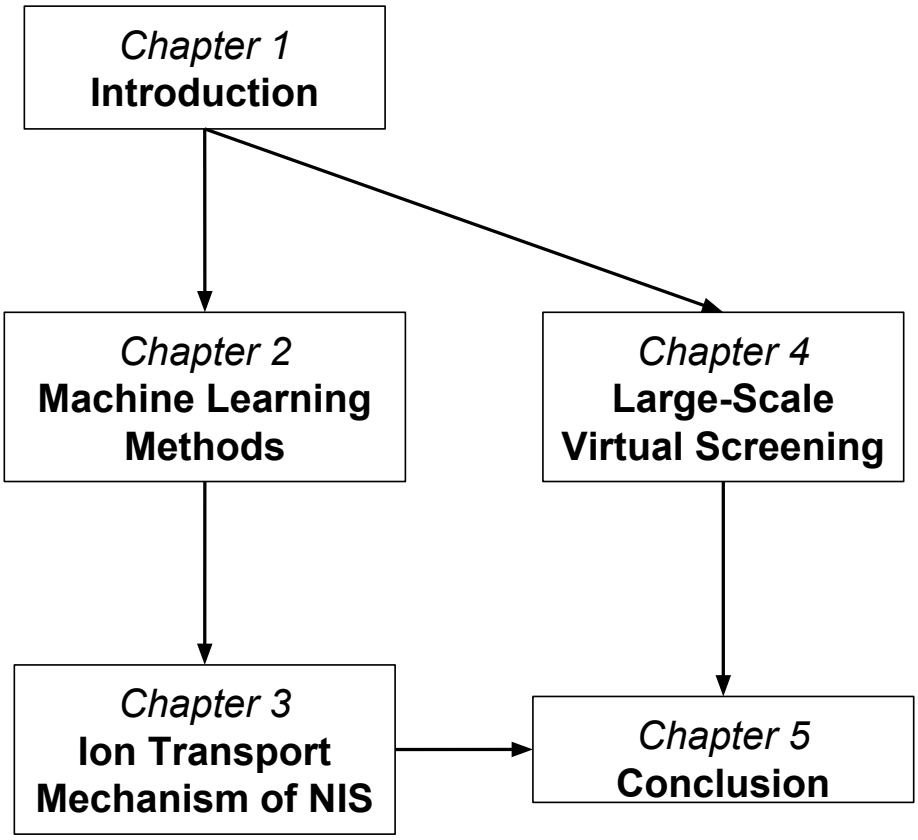

Figure 1.1: Chapter Dependencies

```
//                          # Fiesta3 Root
├── examples                # Examples for using Fiesta3
├── experimental            # Internal Experimental Code
├── script                  # Scripting Programs
│   ├── bash                # Bash Scripting Programs
│   ├── docker              # Internal Docker Build Files
│   ├── php                 # PHP web server programs
│   └── python              # Small Python Applications
├── spark_vina              # Spark Vina
│   ├── collect             # Utilities for Spark Vina
├── cong                    # Cong
├── third_party             # Contains internally maintained source code
│                           # or Bazel Build files of third party softwares
├── tools                   # Bazel Configurations
└── util                    # Shared Utility Libraries
    └── python              # Python Shared Utility Libraries
```

Figure 1.2: Simplified Structure of Fiesta3.

(C++, Python, Golang) in a distributed fashion, Bazel [19] is used as the building tool for Fiesta3. Bazel builds an action graph that demonstrates the dependencies between each file so that it can effectively track changes to the content as well as changes to the build actions. It will launch unrelated actions in two separate threads so as to make full use of distributed machines.

Following Bazel's convention, we use `//` to denote the root folder of Fiesta3. Figure 1.2 shows the organization of Fiesta3.

1.6 MARCC Computational Facility

MARCC is a local supercomputing facility managed jointly by Johns Hopkins University and the University of Maryland, College Park. All our computational tasks were carried out with MARCC. MD simulations got greatly accelerated by graphics processing units (GPU) so that we launched them on GPU nodes, each of which consists of $24 \sim 28$ Intel Xeon Haswell/Broadwell CPU cores and 4 NVIDIA Tesla K80 GPUs. Our clustering task was run majorly with GPU cards and we allocate 4 CPU cores and a GPU per task. Virtual screening jobs were carried out using multiple regular CPU computing nodes (24 Intel Xeon CPU per node).

References

[1] J. Vonck and D. J. Mills. "Advances in high-resolution cryo-EM of oligomeric enzymes". In: *Current Opinion in Structural Biology* 46 (2017), pp. 48–54.

[2] H. Frauenfelder, S. G. Sligar, and P. G. Wolynes. "The Energy Landscapes and Motions of Proteins". In: *Science* 254.5038 (1991), pp. 1598–1603.

[3] F. Mallamace, C. Corsaro, and H. E. Stanley. "Energy Landscape in Protein Folding and Unfolding". In: *P.N.A.S.* 113.12 (2016), pp. 3159–3163.

[4] J. N. Onuchic, Zaida Luthey-Schulten, and P. G. Wolynes. "THEORY OF PROTEIN FOLDING: The Energy Landscape Perspective". In: *Annual Review of Physical Chemistry* 48.1 (1997), pp. 545–600.

[5] C. B. Anfinsen. "Principles that Govern the Folding of Protein Chains". In: *Science* 181.4096 (1973), pp. 223–230.

[6] S. Govindarajan and R. A. Goldstein. "On the thermodynamic hypothesis of protein folding". In: *P.N.A.S.* 95 (1998), pp. 5545–5549.

[7] K Vanommeslaeghe and AD MacKerell. "CHARMM additive and polarizable force fields for biophysics and computer-aided drug design". In: *Biochimica et Biophysica Acta (BBA)-General Subjects* 1850.5 (2015), pp. 861–871.

[8] Kresten Lindorff-Larsen, Stefano Piana, Kim Palmo, Paul Maragakis, John L Klepeis, Ron O Dror, and David E Shaw. "Improved side-chain torsion potentials for the Amber ff99SB protein force field". In: *Proteins: Structure, Function, and Bioinformatics* 78.8 (2010), pp. 1950–1958.

[9] William C Swope, Hans C Andersen, Peter H Berens, and Kent R Wilson. "A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters". In: *The Journal of Chemical Physics* 76.1 (1982), pp. 637–649.

[10] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kale, and K. Schulten. "Scalable molecular dynamics with NAMD". In: *J Comput Chem* 26.16 (2005), pp. 1781–1802.

[11] Mark James Abraham, Teemu Murtola, Roland Schulz, Szilárd Páll, Jeremy C Smith, Berk Hess, and Erik Lindahl. "GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers". In: *SoftwareX* 1 (2015), pp. 19–25.

[12] Christian R Schwantes and Vijay S Pande. "Improvements in Markov state model construction reveal many non-native interactions in the folding of NTL9". In: *Journal of chemical theory and computation* 9.4 (2013), pp. 2000–2009.

[13] Christian R Schwantes and Vijay S Pande. “Modeling molecular kinetics with tICA and the kernel trick”. In: *Journal of chemical theory and computation* 11.2 (2015), pp. 600–608.

[14] Xavier Daura, Karl Gademann, Bernhard Jaun, Dieter Seebach, Wilfred F. van Gunsteren, and Alan E. Mark. “Peptide Folding: When Simulation Meets Experiment.” In: *Angewandte Chemie* DOI: 10.1002/(SICI)1521-3773(19990115)38:1/2¡236::AID-ANIE236¿3.0.CO;2-M (1999).

[15] Robert T McGibbon, Kyle A Beauchamp, Matthew P Harrigan, Christoph Klein, Jason M Swails, Carlos X Hernández, Christian R Schwantes, Lee-Ping Wang, Thomas J Lane, and Vijay S Pande. “MDTraj: a modern open library for the analysis of molecular dynamics trajectories”. In: *Biophysical journal* 109.8 (2015), pp. 1528–1532.

[16] Naveen Michaud-Agrawal, Elizabeth J Denning, Thomas B Woolf, and Oliver Beckstein. “MDAnalysis: a toolkit for the analysis of molecular dynamics simulations”. In: *Journal of computational chemistry* 32.10 (2011), pp. 2319–2327.

[17] Frank Noé. “Probability distributions of molecular observables computed from Markov models”. In: *The Journal of chemical physics* 128.24 (2008), p. 244103.

[18] VS Pande, Kyle Beauchamp, and Gregory R Bowman. “Everything you wanted to know about Markov State Models but were afraid to ask”. In: *Methods* 52.1 (2010), pp. 99–105.

[19] Bazel Team and Open Source Contributors. *Bazel.Build*. URL: `https://www.bazel.build/`.

Chapter 2

Machine Learning Models: Theory and Applications in Analyzing MD Simulations

Machine learning models essentially refer to a palette of diversified mathematical models capable of learning targeted features from large amounts of data and making predictions. These models are designed based on the tasks in question. In the context of MD simulations, since the frames in the trajectories are unlabeled, researchers are majorly interested in accurately identifying data patterns with specific biophysical relevance. Thus, the tasks of interest majorly fall into the category of unsupervised learning, which includes dimensionality reduction, representation learning, clustering, etc. In this chapter, we merely introduce the theory and applications of machine learning models aiming at dimensionality reduction and clustering.

2.1 Notations

The presentation in this chapter and the rest of this dissertation frequently use a few mathematical notations and symbols. To enhance readability and clarify ambiguities, these common symbols and notations are defined in Table 2.1. Readers need to get familiar with the notational conventions we use to read smoothly.

2.2 Dimensionality Reduction

As introduced in Chapter 1, dimensionality reduction is a task of finding optimal lower dimensional representations of data that can approximate the original data the most.

Symbols or Notations	Clarifications and Explanations
$\mathbb{R}$	the set of real numbers
$\mathbb{R}_+$, $\mathbb{R}_{>0}$	the set of positive real numbers
$\mathbb{R}^N$	the set or vector space of N-dimensional real vectors
X	matrices, datasets
$\mathcal{X}$	abstract spaces, topological spaces, linear spaces
x	scalars, one-dimensional random variables
$\mathbf{x}$	column vectors, column random vectors
$\mathbf{x}_i$, x_i	ith element of $\mathbf{x}$
f_θ	functions with a parameter set θ
L^p	L^p norm spaces
$p(\cdot)$, $\mathbb{P}(\cdot)$	probability density functions or probability distributions
$q(\cdot)$	estimated probability densities
$\sum_X$	sum over X
$\mathbb{E}_{\mathbf{z}\sim\mathbb{P}(z)}$	Expectation with respect to $\mathbb{P}(z)$
$\|\cdot\|_p$	p-norm

Table 2.1: Clarifications and explanations on symbols and notations

To express this semantics in mathematics, let's assume that our observed dataset is a sample of a M-dimensional random vector $\mathbf{x}$ with the probability distribution D, dimensionality reduction attempts to find a K-dimensional ($K < M$) representation $\mathbf{y}$ together with a transformation $f : \mathcal{Y} \to \mathcal{X}$ that minimize the reconstruction error. This optimization problem could be generally represented as

$$\min_{f\in\mathcal{H}} \mathbb{E}_{\mathbf{x}} \|\mathbf{x} - f(\mathbf{y})\|_p^q \tag{2.1}$$

where $\mathcal{Y}$ and $\mathcal{X}$ are M and K dimensional vector spaces that $\mathbf{y}$ and $\mathbf{x}$ live in, respectively. And $\mathcal{H}$ represents the Hilbert space where the transformation f lives in. Different dimensionality reduction models confine the scope of searching f in different subsets of $\mathcal{H}$. Moreover, for most of tasks, $\mathcal{X}$ and $\mathcal{Y}$ is merely considered to be $\mathbb{R}^M$ and $\mathbb{R}^K$, respectively. Among all the methods, linear principal component analysis (PCA) is considered as the most widely used and the simplest since it only searches linear embeddings in the K-dimensional subspace $\mathcal{Y}$ of $\mathcal{X}$. In this section, we will briefly introduce Linear PCA as well as two other nonlinear methods and discuss their benefits and caveats.

2.2.1 Linear PCA

Linear (classical) PCA only considers f as linear transformations between vector spaces $\mathbb{R}^K$ and $\mathbb{R}^M$. The following theorem provides a clear view of those linear transformations

and the foundation of the optimization process of PCA.

Theorem 2.2.1. *Let $f \in L(\mathbb{R}^K, \mathbb{R}^M)$. Then minimizing $\mathbb{E}_{\mathbf{x}}[\|\mathbf{x} - f(\mathbf{y})\|_F^2]$ with respect to f is equivalent to minimizing $\mathbb{E}_{\mathbf{x}}[\|\mathbf{x} - U\mathbf{y}\|_F^2]$ with respect to U, where $\|\cdot\|_F$ denotes Frobenius norm and U is a $M \times K$ matrix with K orthogonal columns ($U^T U = I$).*

Proof. For any $f \in L(\mathbb{R}^K, \mathbb{R}^M)$, f has a matrix representation A and A is $M \times K$. A can be factored as $A = UP$ (QR factorization), where U is $M \times K$ with orthogonal columns ($U^T U = I$) and P is $K \times K$. P-factor can be absorbed into $\mathbf{y}$ and be considered as a linear transformation within $\mathbb{R}^K$. Conversely, all matrix operator Us are linear transformations. □

According to this theorem, the general optimization problem (Equation 2.1) is specialized into the following form

$$\min_{U,\mathbf{y}} \mathbb{E}_{\mathbf{x}}[\|\mathbf{x} - U(\mathbf{y})\|_F^2] \tag{2.2}$$

subject to $U^T U = I$. In practice, the expectation over $\mathbf{x}$ could be estimated with an observed, centered dataset represented with a $N \times M$ matrix X. Each row $\mathbf{x}_j$ of X represents a M-dimensional example. Therefore, we replace Equation 2.2 with

$$\min_{U,\mathbf{y}} \sum_{j=1}^{N} \|\mathbf{x}_j - \mathbf{y}_j\|_F^2 = \min_{U,\mathbf{y}} tr(X^T(I_M - UU^T)X) \tag{2.3}$$

Next, we can optimize Equation 2.3 with respect to $\mathbf{y}$ by calculating the stationary point of the Lagrange. This first-step optimization yields $Y = U^T X$. By making this substitution, we can show that Equation 2.3 is equivalent to

$$\min_{U \in \mathbb{R}^{M \times K}} tr(X^T U U^T X) \tag{2.4}$$

subject to $U^T U = I_K$.

Since $U^T X$ is a projection of X on some K-dimensional subspace, this equivalent form (Equation 2.4) can be interpreted quite intuitively. Thus, it claims that miminizing the reconstruction error is equivalent to learning the maximal variance subspace. This form could be further optimized with the same Lagrange multiplier machinery, which

eventually shows that the optimal U should satisfy

$$XX^TU = U\Lambda_K \tag{2.5}$$

where Λ_K is the top-K eigenvalues of XX^T, the covariance matrix and U must be K eigenvectors that correspond to Λ_K. The mechanic derivation of both equations 2.4 and 2.5 are omitted. By following the historical conventions, each eigenvector represents a *principal direction*.

For evaluating the performance of Linear PCA, the proportion of variance explained by the K principal components is calculated by

$$\frac{Var[\mathbf{y}]}{Var[\mathbf{x}]} = \frac{tr(U^TXX^TU)}{tr(XX^T)} = \frac{tr(\Lambda_K)}{tr(\Lambda_M)} \tag{2.6}$$

where the first equation is obtained by definition of the variance and the second by applying U^T to both sides of Equation 2.5.

Linear PCA has an advantage that its results can be simply interpreted since they only involve linear transformations. However, when the data lives on nonlinear manifolds embedded in M-dimensional space, the performance of linear PCA estimated by the proportion of explained variance can be bad. This problem brings the motivation to develop algorithms searching in the sets of nonlinear transformations (Hilbert space) to reduce data dimensionality.

2.2.2 Kernel PCA

If the observed data represented by the current set of features lives on nonlinear manifolds and we still intend to apply the idea of the classical PCA method to reduce the dimensionality of the data, a better set of features has to be constructed from the current set. A natural idea to obtain a better set of representations for all examples in our dataset universally is to map the current representation with a feature map $\phi : \mathcal{X} \to \mathcal{V}$, where $\mathcal{V}$ is a potentially infinite dimensional inner product space (Hilbert space). Please note that $\mathcal{V}$ is required to be inner product space because Equation 2.5 tells us that optimizing the objective function of PCA requires evaluating the covariance matrix in $\mathcal{V}$.

Assuming that there exists a ϕ that can successfully maps $\mathbf{x}$ to some higher or even infinite dimensional space, XX^T can be replaced by a matrix K, whose element $K_{ij} = \widetilde{\phi}(\mathbf{x}_i)^T\widetilde{\phi}(\mathbf{x}_j)$ and $\widetilde{\phi}(\mathbf{x}_i)$ is the centered $\phi(\mathbf{x}_i)$ in $\mathcal{V}$. In practice, calculating inner products in very high dimensional space or Hilbert space is computationally infeasible or impossible. To resolve this issue, the kernel trick plays an important role.

The kernel trick refers to the mathematical fact that for some particular ϕs, the inner product $\widetilde{\phi}(\mathbf{x}_i)^T\widetilde{\phi}(\mathbf{x}_j)$ can be obtained by computing $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ where κ is a well-defined nonlinear function. Instead of computing $\widetilde{\phi}(\mathbf{x}_i)^T\widetilde{\phi}(\mathbf{x}_j)$, $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ is attempted for all i and j. In this case, we can obtain the centered "covariance matrix" in $\mathcal{V}$

$$\widetilde{K} = HKH \tag{2.7}$$

where, H is the centering matrix defined as $= I - \frac{1}{N}1_N 1_N^T$. Because of using kernel tricks, this specific PCA procedure is named as kernel PCA. Kernel PCA can be viewed as an natural extension of classical linear PCA to some extent.

Kernel PCA is not usually used for detecting data patterns in the MD community because of the following cateats:

1. The matrix K has to be computed and maintained in memory, which costs $O(N^2)$ space complexity. When N gets large, it is impossible to hold the matrix in the memory. This problem is a common issue for all kernel methods.

2. Visualization of the projections might be hard to interpret.

3. An appropriate kernel function has to be pre-defined.

Recently, some new landmark kernel methods [1] have been actively developed, which may possibly bring back the attention to kernel PCA.

2.2.3 Deep Autoencoder

Deep autoencoders (DAE) are widely used for unsupervised learning tasks such as learning deep representations or dimensionality reduction. Typically, a traditional deep autoencoder consists of two components, the encoder and the decoder. Each component is

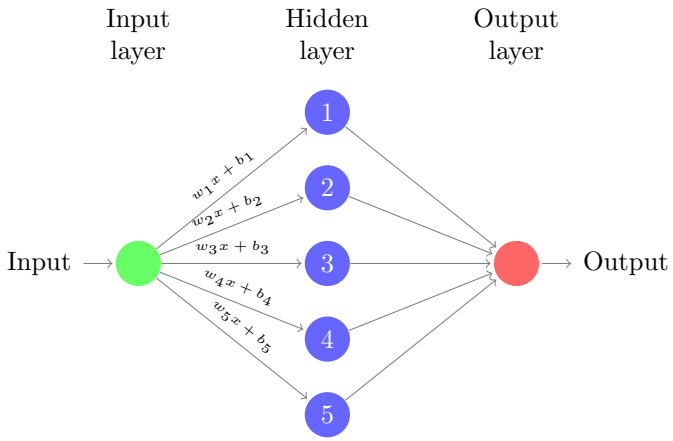

Figure 2.1: A simple example of a feed-forward neural network with one hidden layer.

constructed with a feed-forward neural network, which in essence represents a nonlinear function. A simple example of a feed-forward neural network is shown in Figure 2.1.

Let's denote the encoder's function as $f_\theta : \mathcal{X} \to \mathcal{H}$, and denote the decoder's function as $g_\omega : \mathcal{H} \to \mathcal{X}$, where θ, ω are parameters for each function, $\mathcal{X}$ represents the data space and $\mathcal{H}$ the feature (latent) space. The autoencoder used in the model presented here is always aiming at mapping the high-dimensional data space to a low-dimensional feature space and the quality of the mapping is monitored by how well it can reconstruct the original data space from the feature space. For this purpose, it optimizes the reconstruction loss, which is

$$L(\theta, \omega) = \frac{1}{N} \left\| X - g_\omega(f_\theta(X)) \right\|^2 \tag{2.8}$$

where $L(\theta, \omega)$ represents the loss function for the reconstruction and again, we use a $N \times M$ matrix X to represent our dataset.

The use of the L^2-norm in the reconstruction loss is valid for guiding the reconstructions to remain close to the original data. This approach has to be contrasted with the L^2-norm to measure similarities between frames. In fact, for a finite dimensional vector space V, all l_p norms ($p \geq 1$) are equivalent and induce the same topology. Assuming the dimensionality reduction with DAE is successful, the deep representations generated will feature the major skeleton of MD frames. Therefore, discriminating frames by their deep representations is naturally insensitive to the random noise. Moreover, if we assume a given distribution for the deep representations, the reconstruction function g_ω naturally serves as a generating function that turns the model into a generative model.

Unlike kernel PCA, deep autoencoders learn "the feature maps ϕ" and their "associated kernel functions κ" rather than pre-defining it. This method is completely data-driven. The objective function of DAEs (Equation 2.8) can be optimized by minibatch stochastic gradient descent (SGD) so that no $O(N^2)$ matrices are stored in memory. In addition, if the encoding and decoding functions are restricted to linear transformations, DAEs approach the results of classical Linear PCA. In other words, classical Linear PCA can be viewed as a special case of DAE. To summarize, DAE is a nonlinear,

data-driven method whose objective functions can be easily optimized for dimensionality reduction and representation learning. This method sheds light on data denoising, feature extractions and most importantly, clustering.

In the following chapters, concrete examples of using dimensionality reduction techiniques to problems in MD dynamics are presented.

2.3 Clustering MD Trajectories

2.3.1 Introduction

With the development over the past decade of high performance CPU clusters, GPU accelerated computing and software-level optimization, the performance of general molecular dynamics (MD) software (NAMD[2], GROMACS[3], AMBER[4], CHARMM[5], etc) has been extraordinarily enhanced. Increasingly, individual research groups are capable of running microsecond-scale MD simulations for macromolecular systems of considerable size. Some research laboratories equipped with specialized hardware or large-scale distributed software [6, 7] can even produce millisecond-scale trajectories. Considering an MD trajectory as a number of conformations sampled from the complete ensemble (NVE, NVT or NPT) as it is observed through its time evolution, long trajectories provide significant explorations of the conformational space and could reveal key conformational transitions of the macromolecules. Depending on the simulation, those conformational states may reflect stages in the folding of the protein or details about the mechanism of action of the molecules. Objective unbiased identification of these states is crucial for getting mechanistic information from MD simulations.

Though in many long MD trajectories conformational changes may sometimes be visually recognized, identifying key conformations, grouping similar conformations and quantifying key conformational transitions are obviously more objectively dealt using specialized software. This is because the number of frames in long trajectories is very large and the dimensionality of the simulation system is high. However, this task is often essential for understanding the behavior of the molecules and for comparing changes of their behavior under different environments or those resulting from mutations. For tackling these tasks, similarity-based clustering algorithms, such as KMeans, KMedoids, hiearchical clustering, agglomerative clustering and other domain-specific algorithms are used. Their implementations are widely available as utilities in most major MD packages[3, 8, 9]. All these similarity-based algorithms share the following common basis: clustering procedures depend on frame-wise similarities, usually quantified by

the RMSD. Except for KMeans and KMedoids, which are optimized by expectation maximization, all the other algorithms additionally rely on a pre-determined cutoff for making decisions to branch out clusters.

Similarity-based clustering algorithms solve the clustering task successfully but a few caveats remain. As MD trajectories are getting longer, the number of frames, denoted by N, is growing by orders of magnitude. Since the computation and space complexity of the RMSD matrix are both $O(N^2)$, algorithms based on calculating the RMSD matrix usually take large amounts of computing time and overwhelm the memory. In practice, many researchers run clustering using every n-th frame of the long trajectory; however, this strategy could introduce significant bias into calculations such as those of transition probabilities between clusters. In addition, since RMSDs are essentially Euclidean distances defined on a high dimensional space, clustering using this metric could be vulnerable to artifacts due to their sensitivity to large variations in the conformation of the termini and of flexible loops[10, 11]. Other domain-specific RMSD-based clustering methods may suffer from additional side-effects. For instance, the GROMOS algorithm[12] always concatenates two segments of the time series after extracting a neighborhood as a new cluster, which results in a cutoff-sensitive segmentation of the original trajectory.

In this section, we present two existing popular clustering algorithms that can deal with a large number of frames, KMeans and KMedoids, together with another two new clustering algorithms (see next section) contributed by us. To compare the performance of different algorithms, clustering experiments are done with an extensively long Trp-Cage trajectory and the results of these experiments will be discussed as well.

2.3.2 Theory

2.3.2.1 KMeans and KMedoids

The basic idea of clustering data with KMeans and KMedoids algorithms is to select K perfect cluster centers and assign every data point to a cluster center which is the most similar to itself. The idea can be formalized into the following.

Let's denote our dataset as $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \in \mathbb{R}^M$. The goal is to cluster the dataset into K clusters and the cluster centers of each cluster is denoted as μ_j for $j \in 1, \cdots, N$. For our convenience, we use a **one-hot encoding** to assign membership for any $\mathbf{x}_i$, which is defined as

$$r_i = \mathbf{e}_c \tag{2.9}$$

where $\mathbf{e}_c$ is one of the standard basis of $\mathbb{R}^K$ whose cth element is 1. Equation 2.9 represents that $\mathbf{x}_i$ is assigned to cluster c.

Equipped with these notations, we define an universal objective function for both KMeans and KMedoids as

$$J(r, \mu) = \sum_{i=1}^{N} \sum_{c=1}^{K} r_{ic} \left\| \mathbf{x}_i - \mu_c \right\|_p^m \tag{2.10}$$

When $m = 2$ and $p = 2$, we obtain the concrete objective function for KMeans, while $m = 1$ and $p = 1$ yields the objective for KMedoids. Moreover, KMeans selects μ from the entire $\mathbb{R}^M$, while KMedoids restricts the selection of cluster centeres to elements of dataset. In other words, KMedoids chooses real datapoints as cluster centers.

The general form of J doesn't allow explicit differentiations with respect to r or μ. It implies that we may have to use the expectation maximization (EM) algorithm to optimize the objective rather than gradient-based methods. In addition, since $r \in \{0, 1\}^{N \times K}$ is not a convex set, J is not a convex function. Thus, any algorithm optimizing J is expected to reach a local minimum rather than a global minimum. In practice, this statement is also true for other clustering algorithms with complicated, non-convex objective functions defined.

A popular algorithm, Lloyd's algorithm [13], serves as the skeleton for modern

KMeans implementations. The algorithm is clean and simple. In the first step, it initializes a set of $\{\mu_i\}$ randomly. Then it loops the following steps until convergence.

1. Assign every $\mathbf{x}_i$ to the closest cluster center. (Assignment Step)

2. Compute a new set of $\{\mu_i\}$ based on the current assignments. (Update Step)

The random initialization of $\{\mu_i\}$ is crucial for KMeans and KMedoids to converge quickly and to obtain reasonable clusters. A popular way for initialization, named **k-means++** [14], is commonly used as an effective method to gain full "data coverage". This trick is actually very simple. It picks the very first cluster center uniformly at random and each subsequent cluster center from the rest of the points with probability proportional to its squared Euclidean distance to the closest selected centers. Most of modern KMeans implementations adopts k-means++ as the default option for the initialization.

Practical KMedoids algorithms follow the same structure as Lloyd's algorithm, but they modify the update step with effcient searching strategies. Though all practical KMeans and KMedoids algorithms are not strictly EM algorithms, some of them can be obtained through approaching a limit case of EM for Gaussian mixture models (GMM). [15, 16] Due to this fact, the assumptions made by the limit case ($\pi_i = \frac{1}{K}$ and $\sigma^2 I \to 0$) of GMM[1] can be hardly validated in MD cases, which becomes an implicit caveat of KMeans and KMedoids. Other than that, the similarity metric instability mentioned in the previous section (Section 2.3.1) still applies here. Given these caveats, we are motivated to develop a better clustering method that should have the following properties: 1) the time and space complexity should be proportional to N; 2) since defining an ideal distance metric for measuring similarity between structures is very difficult, the algorithm should be data-driven rather than relying on a pre-defined metric. In other words, ideally, the algorithm would "learn" the metric by optimizing some loss function. 3) The algorithm should always look at the entire dataset to avoid unnecessary segmentations.

[1] GMM is not introduced in this dissertation because it is not applied to any practical MD problems that we have. $\pi_i = \frac{1}{K}$ can be interpreted to mean that all clusters are expected to have same number of examples.

Recently, deep neural network (dNN) models have been proven to be very successful in the field of Artificial Intelligence (AI). Particularly, deep autoencoder models are powerful in developing very complicated non-linear functions that can map a set of raw image pixels in the data space to some separable regions in the feature space and work conversely as generative models. The training process is entirely data-driven and converges after feeding the entire dataset for a limited number of times ("epoches"). Although dNN models are largely employed in AI, they are intrinsically general and can be applied to problems in other fields. Thus, we exploit the data-driven property of dNN, to propose a novel method for clustering MD trajectories based on an existing dNN model – Adversarial Autoencoder (AAE)[17]. In the next sections, we introduce essential building blocks of this novel method.

2.3.2.2 Generative Adversarial Network

Generative Adversarial Network (GAN)[18] is designed to be a deep generative model. It assumes that all observed data are generated from some latent random variable $\mathbf{z}$ with a known prior distribution $p(\mathbf{z})$. The overall goal of GAN is to learn how to generate $\mathbf{x}$ from $\mathbf{z}$, e.g. finding $p(\mathbf{x}|\mathbf{z})$. A GAN model has two components: a generator network (denoted by function $G(\mathbf{z};\theta_g)$) and a discriminator network (denoted by function $D(\mathbf{x};\theta_d)$), which are non-linear functions constructed with multilayer networks. Unlike the autoencoder and many traditional models, GAN is a two-player game rather than an optimizer of a monolithic loss function. The discriminator tries its best to differentiate the real data samples from the generated fake samples, while the generator tries to fool the discriminator by generating counterfeits. When the discriminator is unable to tell the source of the input samples, the generator is considered trained. In order to achieve this goal, GAN trains the discriminator and the generator simultaneously in two phases:

$$\min_G \max_D L(D,G) = \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_{data}(\mathbf{x})}[f(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim\mathbb{P}_{\mathbf{z}}(\mathbf{z})}[f(1-D(G(\mathbf{z})))] \tag{2.11}$$

where $f:\mathbb{R}\to\mathbb{R}$ and $\mathbb{E}[\cdot]$ denotes the expectation of a random variable.

Standard GAN chooses f as the logarithm function and the above adversarial loss

function results in the form of the *Jensen-Shannon* (JS) divergence. Training GAN with the JS divergence loss is very likely to fall into unstable traps such as mode collapse. A recently proposed variant of GAN, Wasserstein GAN (WGAN)[19], replaces the JS divergence loss with Wasserstein loss, which is,

$$W(\mathbb{P}_{data}, \mathbb{P}_{G(\mathbf{z})}) = \sup_{||D||_L \leq K} \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_{data}}[D(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_{G(\mathbf{z})}}[D(\mathbf{x})] \tag{2.12}$$

Essentially, WGAN substitutes f in Equation 2.11 with an identity function and it removes the sigmoid activation function in the last layer of the discriminator network. Empirically, WGAN proved to be more stable but it has to be trained with a fairly small step and weight clipping to keep the discriminator function $D(x)$ approximating a K-Lipschitz function. A recent study[20] shows that the weight clipping leads to optimization difficulties and could result in pathological behaviors. Instead of tuning an optimal clipping threshold, the study suggests enforcing the Lipschitz constraint by adding a gradient penalty term to the original loss function given by Equation 2.12. In our experiments, we train both WGAN components in our model with the gradient penalty (Term $\lambda\mathbb{E}_{\hat{\mathbf{x}}\sim\mathbb{P}_{\hat{\mathbf{x}}}}[(\|\nabla_{\hat{\mathbf{x}}}D(\hat{\mathbf{x}})\|_2 - 1)^2]$ in Equation 2.13) rather than using weight clipping:

$$L(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim\mathbb{P}(z)}[D(G(\mathbf{z}))] - \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_{data}}[D(\mathbf{x})] + \lambda\mathbb{E}_{\hat{\mathbf{x}}\sim\mathbb{P}_{\hat{\mathbf{x}}}}[(\|\nabla_{\hat{\mathbf{x}}}D(\hat{\mathbf{x}})\|_2 - 1)^2] \tag{2.13}$$

where $\hat{\mathbf{x}} = \epsilon\mathbf{x} + (1-\epsilon)G(\mathbf{z})$, and $\epsilon \sim U[0,1]$.

2.3.2.3 Adversarial Autoencoder (AAE)

Generally, AAEs have different kinds of architectures targeted at different tasks. Since our goal is to use it as a generative clustering method, we adopt the unsupervised version of AAE [17]. This version of the architecture (see Figure 2.2) is built based on the data generative process it assumes: each data element $\mathbf{x}$ in the dataset is generated from a latent categorical random variable $\mathbf{y} \sim Cat(\boldsymbol{\pi})$, represented as a K-dimensional one-hot vector (K is the dimension of $\mathbf{y}$ as well as a user-defined cluster number) and a Gaussian style variable $\mathbf{z} \sim N(0, I)$. The autoencoder component of the AAE estimates the encoding function $p(\mathbf{y}, \mathbf{z}|\mathbf{x})$ and the decoding function $p(\mathbf{x}|\mathbf{y}, \mathbf{z})$ by minimizing the

reconstruction loss. Here we denote the encoding function and decoding function as $p(\cdot)$, since in general, $p(\cdot)$ could be modeled as either deterministic functions or probabilistic densities. However, in this thesis presentation [17], we model these functions as deterministic. Following the original paper, we denote the estimate of $p(\cdot)$ as $q(\cdot)$. While optimizing the reconstruction loss, the category GAN and the style GAN regulate the autoencoder's encoder by imposing the prior distribution of $\mathbf{y}$ and $\mathbf{z}$ onto $q(\mathbf{y})$ and $q(\mathbf{z})$, where $q(\cdot)$ is defined as,

$$q(\cdot) = \int_X q(\cdot|x)\mathbb{P}(\mathbf{x})d\mathbf{x} \tag{2.14}$$

With these considerations, we can write the reconstruction loss as,

$$L = \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathbb{P}(\mathbf{x})} \mathop{\mathbb{E}}_{\substack{\mathbf{y}\sim q_{cat}(\mathbf{y}) \\ \mathbf{z}\sim q_{style}(\mathbf{z})}} \|g_s(g_{cat}(\mathbf{y}) + g_{style}(\mathbf{z})) - \mathbf{x}\|^2 \tag{2.15}$$

where g_s, g_{cat} and g_{style} denote the decoding function of the shared decoder, the categorical decoder and the style decoder, respectively.

Intuitively, the autoencoder guards the quality of the latent features with its reconstruction loss. The category GAN pushes the categorical feature vector to K-simplex and the style GAN leads $p(\mathbf{z})$ to a standard Gaussian. In other words, AAE assigns an expected $N\pi_i$ number of frames into the ith bucket and each bucket has a standard Gaussian shape. Based on the fact that AAE can impose prior distributions to latent representations successfully, it can feed samples from the prior distributions to the trained generative decoder to create new conformations that do not exist in the original trajectories.

A serious drawback of this method is that it needs to have a correct expectation of the categorical mass π, which is usually impossible to estimate in practice. Intuitively, imposing an incorrect prior knowledge onto the categorical encodings may harm the reconstruction loss, and therefore the quality of the clustering. It remains hard to mathematically estimate the penalty caused by the incorrect prior due to the complex nature of the optimization of this model. A "work-around" solution to this issue is to define a large number of small clusters and assign a uniform distribution to them. This solution can remove the potential penalty to some extent. However, additional analytical

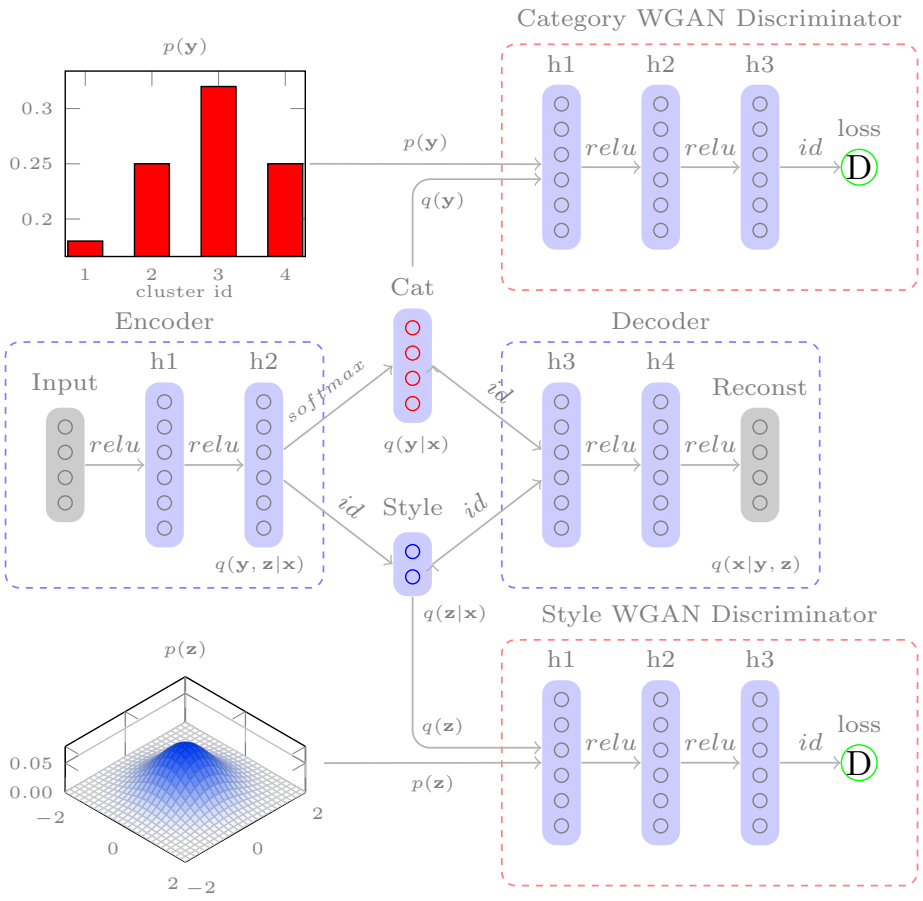

Figure 2.2: A general architecture of the unsupervised version of AAE. The architecture consists of four major components: the encoder, the decoder, the discriminator for categorical GAN and the discriminator for style GAN. Each component is represented as a neural network, which is represented as a multilayer perceptron in this figure. A general choice of activation functions applied to all layers are marked above the arrows connecting all layers. "relu" is short for Rectifier Linear Unit (ReLU) and "id" is short for the identity function. "h1", "h2", etc represent hidden layers in a neural network component. The middle row of the architecture essentially forms a deep autoencoder whose deep representation is divided into a categorical part (red circles) and a style part (blue circles). The encoder combined with the top row forms the categorical GAN and with the bottom row forms the style GAN. $p(\mathbf{y})$ illustrates a categorical distribution and $p(\mathbf{z})$ illustrates a standard Gaussian.

steps are needed for further grouping the resulting small clusters and, specifically in MD cases, for understanding the estimated transition probability matrix.

Our proposal is the reparameterization of the categorical prior with a Gumbel-Softmax distribution [21, 22] to eliminate the need to specify the categorical mass beforehand. In other words, we propose a special AAE model for clustering that does not need any prior knowledge of how examples are distributed among clusters.

2.3.2.4 Reparameterization Trick

Reparameterization Trick refers to a sampling process in the machine learning and statistics world. The major aim of using the reparametrization trick is to separate the stochastic sampling part and parameter-dependent transformation part of a parameterized distribution. In stricter terms, it assumes that there is a parameterized distribution $D(\alpha)$, where α is the parameter. To reparameterize $D(\alpha)$, we first sample from a unparameterized distribution Z and then apply a deterministic function that depends on the parameter $f_\alpha(\mathbf{z})$ to the samples to make them as directly sampled from $D(\alpha)$.

After uncoupling the parameters from the stochastic sampling, the distribution parameters become trainable model parameters since their gradients can be estimated by calculating the partial derivative of the loss function with respect to them. In our case, we rewrite the reconstruction loss by reparameterizing the categorical prior,

$$L = \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathbb{P}(\mathbf{x})} \mathop{\mathbb{E}}_{\substack{\boldsymbol{\alpha}\sim q_{cat}(\boldsymbol{\alpha}) \\ \mathbf{z}\sim q_{style}(\mathbf{z})}} f(h(\boldsymbol{\alpha}, \boldsymbol{\pi}), \mathbf{z}) \tag{2.16}$$

where $f(\mathbf{y}, \mathbf{z}) = \|g_s(g_{cat}(\mathbf{y}) + g_{style}(\mathbf{z})) - \mathbf{x}\|^2$ and we use $h(\boldsymbol{\alpha}, \boldsymbol{\pi})$ to substitute $\mathbf{y}$. In addition, we use $\boldsymbol{\alpha}$ to represent a random vector with a fixed distribution and $\boldsymbol{\pi}$ to represent the categorical mass. The function $h(\cdot)$ is the transformation function.

Based on Equation 7, we can obtain the gradient with respect to $\boldsymbol{\pi}$,

$$\nabla_{\boldsymbol{\pi}} L = \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathbb{P}(\mathbf{x})} \mathop{\mathbb{E}}_{\substack{\boldsymbol{\alpha}\sim q_{cat}(\boldsymbol{\alpha}) \\ \mathbf{z}\sim q_{style}(\mathbf{z})}} \nabla_h f(h(\boldsymbol{\alpha}, \boldsymbol{\pi}), \mathbf{z}) \nabla_{\boldsymbol{\pi}} h(\boldsymbol{\alpha}, \boldsymbol{\pi}) \tag{2.17}$$

The *Gumbel-Max Trick*[23, 24] is a reparameterization of a one-hot categorical distribution. It samples a d-dimensional vector from a standard Gumbel distribution $\boldsymbol{\alpha}$ and uses $f(\boldsymbol{\alpha}) = \text{onehot}(\text{argmax}(\boldsymbol{\alpha} + \log \boldsymbol{\pi}))$ as the transformation function. Since the argmax function is not a differentiable function, the gradient cannot propagate to the parameter $\boldsymbol{\pi}$. The Gumbel-Softmax distribution aims to relax the Gumbel-Max Trick by replacing the argmax function with softmax.

$$y_i = \frac{\exp\left((\log \pi_i + \alpha_i)/\tau\right)}{\sum_{j=1}^{k} \exp\left((\log \pi_j + \alpha_j)/\tau\right)} \tag{2.18}$$

This distribution was originally introduced as Concrete Distribution by Maddison et al.[21] and was successfully used in a related work[22]. The distribution also relieves the constraint that the categorical mass should belong to the $(k-1)$-dimensional simplex. π_is are not necessarily normalized, so they can be anywhere on $\mathbb{R}_+$. When directly learning the logits of π_is, no constraint is needed since the logits belong to $\mathbb{R}$. The extra parameter τ grants the distribution a few interesting properties, illustrated in Theorem 2.3.2. To prove these properties, let's first prove a lemma that shows the probability of get a maximal value at position k' when sampling from a Gumbel-softmax distribution.

Lemma 2.3.1. *Let* $\mathbf{g}$ *be a* d*-dimensional random vector and all its components are independent identical distributed (i.i.d) random variables with a Gumbel* $(0, 1)$ *distribution.* $\boldsymbol{\pi}$ *be a* d*-dimensional vector in* $(0, +\infty)^d$ *and* τ *is an arbitrary number in* $R_{>0}$*. Let's define a* d*-dimensional random vector* $\mathbf{y}$ *by transforming* $\mathbf{g}$ *with function* $\mathbf{f}$*, where* $\mathbf{f}$*:* $R^d \to R^d$*. The* k*th element of* $\mathbf{f}$ *is*

$$y_k = f_k(\mathbf{g}) = \frac{\exp\left((\log\left(\pi_k\right) + g_k)/\tau\right)}{\sum_k \exp\left((\log\left(\pi_k\right) + g_k)/\tau\right)} \tag{2.19}$$

where y_k*,* g_k *and* π_k *are the* k*th element of* $\mathbf{y}$*,* $\mathbf{g}$ *and* $\boldsymbol{\pi}$*, respectively. Then we have,* $\forall k' \in \{1, \ldots, d\}$*,*

$$p(y_{k'} = \max_k y_k \mid \boldsymbol{\pi}, \tau) = \frac{\pi_{k'}}{\sum_k \pi_k} \tag{2.20}$$

Proof. Let's first consider the conditional probability of $y_{k'} = \max_k y_k$ given $\boldsymbol{\pi}$, τ and

$g_{k'} = g$. Since g_ks are i.i.d. and $y_{k'}$ is the maximum among all elements of $\mathbf{y}$, we have,

$$\begin{aligned} p(y_{k'} = \max_k y_k \mid \boldsymbol{\pi}, \tau, g_{k'} = g) &= \prod_{k \neq k'} p(y_k \leq y_{k'}) \\ &= \prod_{k \neq k'} p(\log \pi_k + g_k \leq \log \pi_{k'} + g) \\ &= \prod_{k \neq k'} p(g_k \leq \frac{\pi_{k'}}{\pi_k} + g) \end{aligned} \tag{2.21}$$

Since $g_k \sim \text{Gumbel}(0, 1)$ and the cumulative distribution function of Gumbel(0, 1) is $e^{-e^{-x}}$, $x \in \mathbb{R}$, we substitute $p(g_k < \frac{\pi_{k'}}{\pi_k} + g)$ with $\exp\left(-\exp\left(-\log\left(\frac{\pi_{k'}}{\pi_k}\right) - g\right)\right)$. Then we have,

$$\begin{aligned} p(y_{k'} = \max_k y_k \mid \boldsymbol{\pi}, \tau, g_{k'} = g) &= \prod_{k \neq k'} \exp\left(-\exp\left(-\log\left(\frac{\pi_{k'}}{\pi_k}\right) - g\right)\right) \\ &= \exp \sum_{k \neq k'} \left(-\exp\left(-\log\left(\frac{\pi_{k'}}{\pi_k}\right) - g\right)\right) \\ &= \exp \sum_{k \neq k'} \left(-\frac{\pi_k}{\pi_{k'}} e^{-g}\right) \\ &= \exp\left(-e^{-g} \frac{\sum_{k \neq k'} \pi_k}{\pi_{k'}}\right) \end{aligned} \tag{2.22}$$

The probability density function of Gumbel(0, 1) is $e^{-x-e^{-x}}$, where $x \in \mathbb{R}$. Marginalize out the $g_{k'}$, we have,

$$\begin{aligned} p(y_{k'} = \max_k y_k \mid \boldsymbol{\pi}, \tau) &= \int_{g_{k'}} p(y_{k'} = \max_k y_k \mid \boldsymbol{\pi}, \tau, g_{k'} = g) p_{g_{k'}}(g) dg \\ &= \int_{-\infty}^{+\infty} \exp\left(-e^{-g} \frac{\sum_{k \neq k'} \pi_k}{\pi_{k'}}\right) e^{-g-e^{-g}} dg \\ &= \int_{-\infty}^{+\infty} \exp\left(-g - e^{-g} \frac{\sum_k \pi_k}{\pi_{k'}}\right) dg \end{aligned} \tag{2.23}$$

Let $C = \frac{\sum_k \pi_k}{\pi_{k'}}$, the improper integral in the last step of Equation 2.23 can be rewritten as $\int_{-\infty}^{+\infty} \exp\left(-g - Ce^{-g}\right) dg$.

This integral can be easily calculated by substituting g with $-\log t$. We have,

$$\begin{aligned}\int_{-\infty}^{+\infty} \exp\left(-g - Ce^{-g}\right) dg &= \int_{0}^{+\infty} e^{-Ct} dt = \\ &= -\frac{1}{C} e^{-ct}\Big|_{0}^{+\infty} \\ &= \frac{1}{C}\end{aligned} \tag{2.24}$$

Subsituting C back to $\frac{\sum_k \pi_k}{\pi_{k'}}$ completes the proof. □

Theorem 2.3.2. *Let $\boldsymbol{y}$ be a random variable sampled from Equation 2.19, where $\boldsymbol{\pi} \in (0, +\infty)^d$, $\tau \in (0, +\infty)$, then we have,*

(1) $\forall k$, $\lim_{\tau \to +\infty} y_k = \frac{1}{d}$. *In fact,* $\lim_{\tau \to +\infty} y_k$ *defines a random variable, which degenerates to a constant* $\frac{1}{d}$ *everywhere in the event space* Ω*, which is* $\mathbb{R}^k$.

(2) $\lim_{\tau \to 0^+} \mathbf{y} \sim$ *OneHotCategorical*$(\frac{\boldsymbol{\pi}}{\sum_{i=1}^k \pi_i})$

(3) $\tau \to 0^+$, $\mathbb{E}y_i = \frac{\pi_i}{\sum_{j=1}^k \pi_j}$.

Proof. (1) $\forall \omega \in \Omega$, $\forall k \in \{1, \ldots, d\}$, $|g_k| < +\infty$, so as $|\log(\pi_k) + g_k| < +\infty$. In this case, when $\tau \to +\infty$, $(\log(\pi_k) + g_k)/\tau \to 0$ and $\exp((\log(\pi_k) + g_k)/\tau) \to 1$. It is trivial to see $y_k = \frac{1}{d}$.

(2) Since $\mathbf{g}$ is a random vector defined on the event space $\mathbb{R}^d$, which is denoted as Ω in the following text, $\mathbf{y} = \mathbf{f} \circ \mathbf{g}$ is also defined on Ω. Let's start by defining a partition of Ω.

Consider a series of subsets of Ω, denoted by $\{A_k\}$ where k is the index. Each A_k is defined as $\{\omega \mid \omega \in \Omega, k \in I_{max}, |I_{max}| = 1\}$, where $I_{max} = \arg\max_k(g_k(\omega) + \log(\pi_k))$ and $|\cdot|$ denotes the cardinality. Let's define another set $A_{>1}$ as $\{\omega \mid \omega \in \Omega, |I_{max}| > 1\}$. Since $\mathbf{g}$ is a d-dimensional random vector, $\forall \omega$, $|I_{max}| \leq d$. In addition, since $\forall \omega$, each component of $\mathbf{g}(\omega)$ is finite and bounded, then there must be at least one maximum. This implies $|I_{max}| \geq 1$, $\forall \omega$. Therefore, it is trivial to see $(\cup_k A_k) \cup A_{>1}$ is a partition of Ω.

Let's consider the value of $\lim_{\tau \to 0^+} y_k$ on each A_k. $\forall \omega \in A_k$, by definition, we have, for $\forall j,\ j \in \{1, \ldots, d\}$,

$$\begin{aligned} y_j = f_j(\mathbf{g}(\omega)) &= \frac{\exp\left((\log(\pi_j) + g_j)/\tau\right)}{\sum_i \exp\left((\log(\pi_i) + g_i)/\tau\right)} \\ &= \frac{\exp\left((\log(\pi_j) + g_j - \log(\pi_k) - g_k)/\tau\right)}{\sum_i \exp\left((\log(\pi_i) + g_i - \log(\pi_k) - g_k)/\tau\right)} \end{aligned} \tag{2.25}$$

The last step of Equation 2.25 is obtained by dividing the term $\exp(\log(\pi_k) + g_k))$ from both the numerator and the denominator. It is trivial to see that for any $j \neq k$, $\lim_{\tau \to 0^+} \exp\left((\log(\pi_j) + g_j - \log(\pi_k) - g_k)/\tau\right) = 0$ and only when $j = k$, the limit is equal to 1. This is also true for i in the denominator. Then we can conclude, for any index k,

$$\forall \omega \in A_k,\ \lim_{\tau \to 0^+} y_j = 1 \text{ when } j = k, \text{ otherwise } 0. \tag{2.26}$$

Similarly as the above analysis, we can draw the corresponding conclusion for set $A_{>1}$.

$$\forall \omega \in A_{>1},\ \lim_{\tau \to 0^+} y_j = \frac{1}{|I_{max}|} \text{ when } j \in I_{max}, \text{ otherwise } 0. \tag{2.27}$$

For simplicity, we use $\mathbf{e}_k$ to denote the unit vector on k-th dimension in $\mathbb{R}^d$, which is defined as only the k-th element is 1 and all others are 0. And we denote the set that contains all values of $\lim_{\tau \to 0^+} \mathbf{y}$ in Equation 2.27 as O. Then Equation 2.26 implies $p(A_k) \leq p(\lim_{\tau \to 0^+} \mathbf{y} = \mathbf{e}_k)$ and Equation 2.27 implies $p(A_{>1}) \leq p(O)$.

It is also trivial to see that $\lim_{\tau \to 0^+} \mathbf{y}(\omega) = \mathbf{e}_k$ implies $\omega \in A_k$. This can be proved by contradiction. Assume $\omega \notin A_k$, since $(\cup_k A_k) \cup A_{>1}$ is a partition of Ω, ω must be in any other set A_j where $j \neq k$ or $A_{>1}$. Apparently this is impossible because of Equation 2.26 and 2.27. Then we have $p(\lim_{\tau \to 0^+} \mathbf{y} = \mathbf{e}_k) \leq p(A_k)$ and $p(O) \leq p(A_{>1})$.

$$\begin{aligned} p(A_k) &= p(\lim_{\tau \to 0^+} \mathbf{y} = \mathbf{e}_k) \\ p(A_{>1}) &= p(O) \end{aligned} \tag{2.28}$$

Due to Lemma 2.3.1 and its proof, we know $p(A_k) = \frac{\pi_k}{\sum_i \pi_i}$. (Please notice that

Gumbel(0, 1) is a continuous distribution, so $p(y_k \leq g_{k'}) = p(y_k < g_{k'})$ in the proof of Lemma 2.3.1.) By summing k, we have $\sum_k p(A_k) = 1$. On the other hand, because of the fact that $(\cup_k A_k) \cup A_{>1}$ is a partition of Ω, $\sum_k p(A_k) + p(A_{>1}) = 1$. Then we get $p(A_{>1}) = 0$, which makes the points in $A_{>1}$ and O unmeaningful. (This can be seen in another way that $A_{>1}$ is exactly a zero-measure set.) It turns out that $\lim_{\tau \to 0^+} \mathbf{y}$ is defined only on $\{\mathbf{e}_k\}$. Take $\{\mathbf{e}_k\}$ as Ω' and $\mathcal{F}'$ is the σ-algebra on it. Combining with the result in Equation 2.28, we get the distribution of $\lim_{\tau \to 0^+} \mathbf{y}$ as,

$$p'(\lim_{\tau \to 0^+} \mathbf{y} = \mathbf{e}_k) = \frac{\pi_k}{\sum_i \pi_i} \tag{2.29}$$

which is usually written as

$$p'(\lim_{\tau \to 0^+} \mathbf{y}) = \prod_j (\frac{\pi_j}{\sum_i \pi_i})^{y_j} \tag{2.30}$$

where p' is the probability measure defined on the probability space $(\Omega', \mathcal{F}', p')$.

(3) This is a trivial corollary of (2).

□

2.3.2.5 AAE with Gumbel-Softmax

A general architecture of our model, namely, AAE with Gumbel-Softmax (AAE-GS), is illustrated in Figure 2.3. By reparameterizing the categorical prior with the Gumbel-Softmax distribution, we extract the categorical mass as a trainable variable, which enables us to impose only the "stochastic" part of the categorical distribution. In this case, the logits of the categorical mass will be updated each time the autoencoder's loss is updated.

To train the model, instead of assuming a categorical mass, we define an annealing process of the parameter τ in Equation (2.19). First, we set τ to a large value to simulate $\tau \to +\infty$. By Theorem 2.3.2, we know the model will approximately put every example in only one cluster, since the categorical representation for each example is almost same. We then anneal τ to a value close to zero in some finite number of steps so that eventually the model can assign each example a one-hot vector marking the most

suitable bucket to put it in.

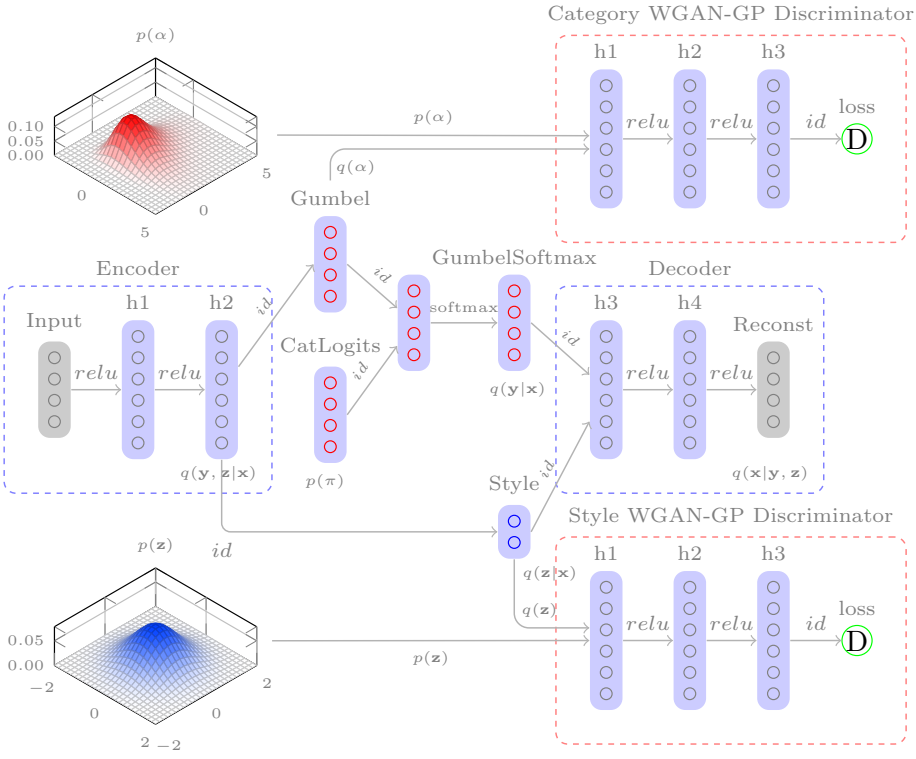

Figure 2.3: A general architecture of AAE with Gumbel-softmax. The architecture is constructed similarly as AAE. However, $p(\alpha)$ illustrates a Gumbel(0, I) distribution, which combines the variable $p(\pi)$ to form the categorical representation.

2.3.3 Experiments and Results

2.3.3.1 A Testing Model for Clustering Algorithms: Trp-Cage

Unfortunately, a standard dataset for examining the clustering performance of different algorithms for MD frames has not been established yet. To test our algorithms, a model trajectory with the following properties was required: 1) It samples *a small protein*, 2) It is a long enough simulation that traverses a large scope of the protein's conformation space, 3) It samples conformations of the protein that are easily distinguishable by the human eyes.

A few years ago, Lindorff-Larsen et al. [25] did a thorough MD based folding study on 12 small proteins with pure MD simulations. It was observed that each protein folded to a conformation very close to its experimentally determined structure during those simulations, and that the process of folding occurred many times during the simulation. There were 9 fast-folding proteins amongst the 12 selected by Lindorff-Larsen et al. We picked a small protein, *Trp-cage* (PDB ID: 2JOF) from these 9 fast-folders. Trp-cage is a 20-residue protein that folds as two small α-helices and a short loop. The fold is stabilized by having a tryptophan residue (Trp6) at the core of the protein with its indole side chain sandwiched between two prolines (Pro12 and Pro18) and surrounded on the other sides by a tyrosine (Tyr3, a leucine (leu7) and another proline (Pro19). This protein has a folding time of 14 μs and was simulated for a total of 208 μs. Compared to other fast-folding proteins provided by the study, the folded structure of Trp-cage has clear secondary structure elements that can fold into different conformations even though its length is relatively small. Additionally, its short folding time makes the size of the trajectory compact. For our purposes, this protein is an ideal model for testing the clustering algorithms. The trajectory was generously provided by *D.E. Shaw Research*.

The trajectory we obtained contains a total of 1,044,000 frames assembled in 105 shards. In our clustering experiments, we only consider the backbone coordinates of the protein. Therefore, we sliced each frame by the backbone selection. The selection includes 80 atoms of Trp-Cage so that flattening all the coordinates per frame of these

atoms results in a 240-dimensional array. The full trajectory was aligned to a specific frame and translated into TFRecords format specifically designed for Tensorflow[26] Framework and resharded into 10 shards. We use this dataset for all our experiments.

2.3.3.2 Clustering with KMedoids

Before we experiment on the clustering methods we proposed, a clustering experiment on our dataset using KMedoids was performed. To be unbiased, we use the open source implementation of KMedoids (`MiniBatchKMedoids`) offered in the MSMBuilder package [27]. In this experiment, we cluster the trajectories into 8 clusters with RMSD as our similarity metric. We allow the KMedoids instance running 500 iterations at maximum. The implementation of this experiment is available at `// notebooks/ClusteringWithKMeansAndPlot.ipynb`. This experiment takes about 12 hours to complete on MARCC.

Figure 2.4 shows the result of this experiment. The functionality of sorting frames based on their similarities to the cluster center is not directly available in MSMBuilder's implementation so that we simply select representative frames in each cluster randomly.The probability mass for almost all clusters (except cluster 1 and 7) approaches 1/8. This implies KMedoids attempts to put equal number of frames in every cluster if possible, which can be a direct influence from the assumptions KMeans-like algorithms made if they are considered as limit cases of GMM. This characteristic of the algorithm potentially yields duplicated clusters (cluster 0 and cluster 4). From the visualization of the results, we found KMedoids can successfully identify some featured conformations, such as the folded conformation (cluster 0 and cluster 4) and the unfolded conformation (cluster 1) with a few exceptions. However, for some of the clusters (cluster 2), key features are not recognizable. Though the result of this algorithm shows a few misclassifications, it can definitely serve as a control baseline for other experiments.

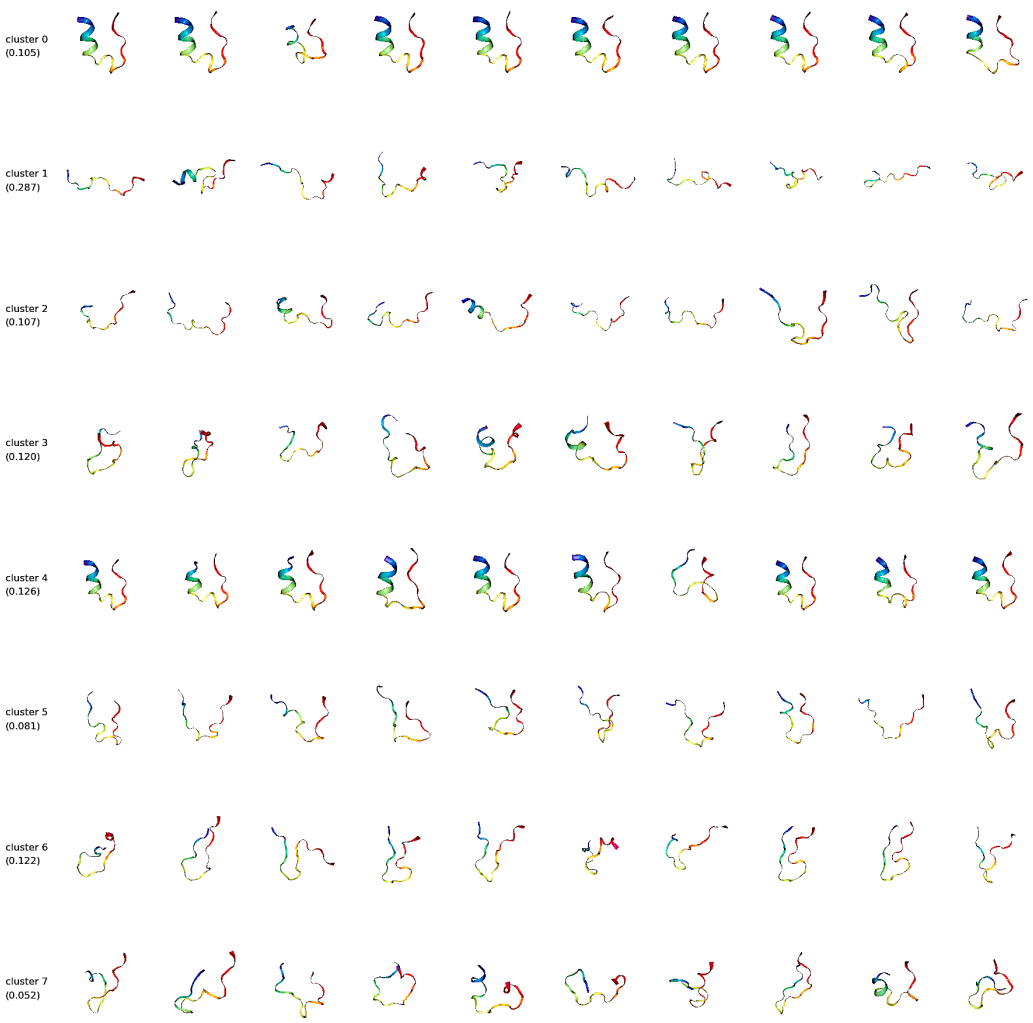

Figure 2.4: Visualization of cluster representatives obtained from KMedoids experiment. Ten frames randomly selected from each cluster are shown as the representatives in each row. All structures are oriented to a fixed orientation. The estimated probability mass for each cluster is listed in brackets underneath each cluster id label.

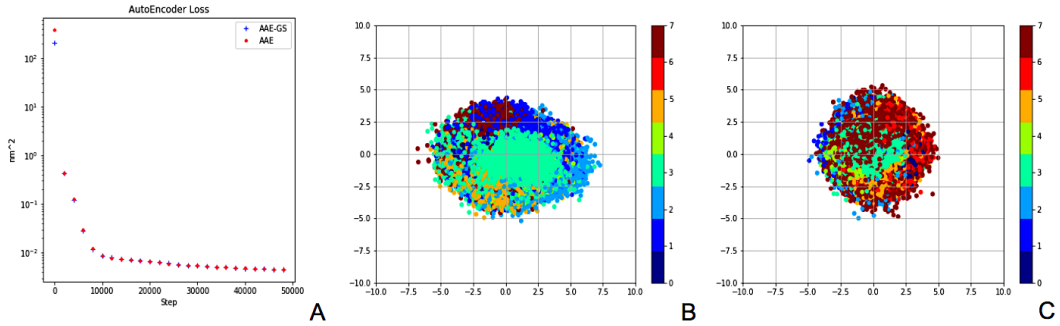

Figure 2.5: Convergence of AAE and AAE-GS clustering experiments. The restruction losses plotted in A of both experiments were evaluated every 2000 steps. B and C plots the first two dimensions of the "learnt" style vectors of all frames in AAE and AAE-GS experiments, respectively.

Table 2.2: Structures of all sub-components of our AAE model.

Component	Architecture[2]
Shared Encoder	$400_{\text{BN, 0.1 DO}} \times 400_{\text{BN}} \times 400_{\text{BN}}$ [3]
Categorical Encoder	$100_{\text{BN}} \times 8^{\text{softmax}}$
Style Encoder	$200_{\text{BN}} \times 16^{\text{id}}$
Categorical Decoder	$100_{\text{0.1 DO}} \times 400_{\text{BN}}$
Style Decoder	$200_{\text{0.1 DO}} \times 400_{\text{BN}}$
Shared Decoder	$400_{\text{BN, 0.1 DO}} \times 400_{\text{BN}} \times 240^{\text{id}}_{\text{BN}}$
Discriminator (Cat-GAN)	$400_{\text{BN, 0.1 DO}} \times 400_{\text{BN, 0.1 DO}} \times 400 \times 200 \times 200 \times 200 \times 200 \times 1^{\text{id}}$
Discriminator (Style-GAN)	$400_{\text{BN, 0.1 DO}} \times 400_{\text{BN, 0.1 DO}} \times 400 \times 200 \times 200 \times 200 \times 200 \times 1^{\text{id}}$

2.3.3.3 Clustering with AAE

We performed a clustering experiment on our dataset using the general unsupervised architecture of AAE. We assumed the categorical prior obeys the discrete uniform distribution and used 8 as the number of clusters. We set the dimension of the style representation to 16 and all its components as independent identically distributed (i.i.d) standard Gaussians. The hyperparameters and the architecture we used for this experiment are listed in Table 2.2. We trained the model until the autoencoder's loss converged, which took $>$ 24 hours on one NVIDIA Tesla K80 GPU card. (See Figure 2.5 A). The final reconstruction loss was 0.005 nm^2, so the average difference between the coordinate of any atom and that in its reconstruction of that atom is approximately 0.7 Å.

To better visualize frames in each cluster, we sorted the frames in each cluster by the norm of their style vectors from small to large, since we consider that the smaller the norm of the noise the more representative a frame is (Figure 2.6).

The results show that the uniform categorical prior is successfully imposed on the categorical representation $\mathbf{y}$. The first two dimensions of the style representation for all frames are plotted in Figure 2.5 B, which shows that the style representation obeys a Gaussian distribution. From the visualization of frames in every cluster, we found that the algorithm identified several featured conformations, for example, cluster 0, 5, 6 and 7 being identified as well-packed conformations, cluster 4 as the unfolded conformation and cluster 3 as some interesting intermediate state on way to fold. We notice that the frames in cluster 0, 5, and 6 are quite similar and could be treated as duplicate clusters.

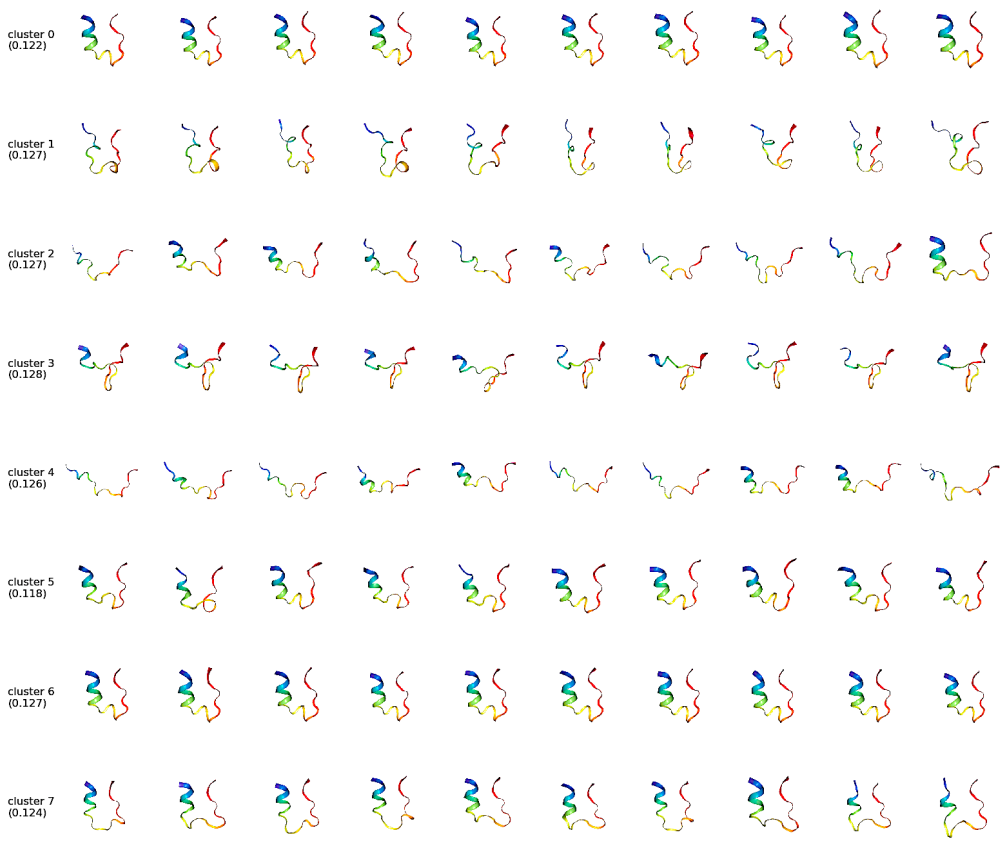

Figure 2.6: Visualization of cluster representatives obtained from AAE experiment. The first ten frames of each cluster sorted by the norm of the style vector are shown in each row. All structures are oriented to a fixed orientation. The estimated probability mass for each cluster is listed in brackets underneath each cluster id label.

Intuitively, this makes sense because the folded structure has a greater population among all states since in the conditions of the simulation, it is energetically stable but we forced every cluster contain the same number of frames.

2.3.3.4 Clustering with AAE-GS

Using the same assumption about the distribution of the style vector and its dimension as in the previous experiment, we did a clustering experiment using AAE with Gumbel-Softmax. We use a similar architecture as the previous experiment except that we included the reparametrization with Gumbel-Softmax (See Table 2.2). Based on the model of AAE-GS, we define an annealing process of the parameter τ (See Figure 2.7) to cool down the categorical representation from a uniform vector to a one-hot vector. We trained the model until convergence. AAE-GS takes similar amount of time as the AAE experiment by using a similar model architecture. The convergence of the reconstruction loss and the successful imposition of the Gaussian noise are shown in Figure 2.5 A and C.

This experiment shows that the folded structures are clustered into only one cluster (cluster 3) instead of three duplicated clusters obtained in the AAE experiment. Cluster 3 contains frames with the same characteristics of the folded structure: Trp6 is caged by residues Pro12, Pro18, Tyr3, Leu7 and Pro19. (See Figure 2.10) The probability mass for this folded cluster is approximately twice the mass of the same cluster in the AAE experiment. It seems that the AAE-GS algorithm tends to aggregate all folded structures within those duplicate clusters into one. However, the probability mass was doubled, not tripled because the structures in a AAE cluster may not be strictly folded as the norm of the style vector gets larger. The algorithm can categorize those loosely packed structures into other clusters. By comparing all 8 clusters in Figure 2.8, we found there are no obvious duplicated clusters.

2.3.3.5 Generation of fake frames

As explained in the Theory section, with a trained decoding function and latent variables with known distributions, we can generate fake frames that do not exist in the original

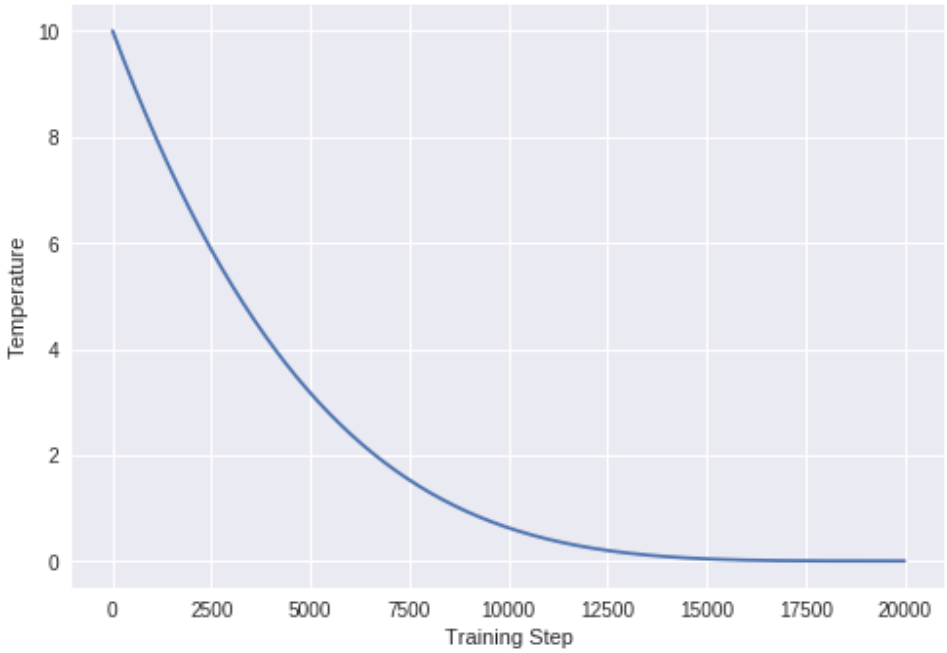

Figure 2.7: The annealing process of the parameter τ.

Figure 2.8: Visualization of cluster representatives obtained from AAE-GS experiment.

trajectory. Figure 2.9 shows a few fake frame examples generated from cluster 4 using the AAE-GS model. The sampled style vector used to generate the fake frame is shown below the structure.

The reconstruction quality could be furthered improved by optimizing the hyper-parameters of the model components. For instance, we could add Convolution Neural Networks (CNN) to our model to increase the representation power of all components, which has proven to be very effective in computer vision tasks. Generation of fake frames can enrich the existing dynamics and could be useful for searching for new conformations with specific biophysical relevance.

2.3.4 An application of clustering frames with AAE-GS

To evaluate the significance of the clusters identified in the long Trp-Cage MD trajectory by our AAE-GS computation we assumed that they may represent significant intermediates in the folding of the small protein. Several groups have studied the folding of Trp-Cage using MD simulations [28–32]. Some of these studies used clustering and other algorithms to infer a possible folding pathway. In our clustering, it is clear that cluster 0 corresponds to the unfolded state (U) and that cluster 3 corresponds to the fully folded protein (F). The significance of these and the other clusters was explored by using all the clusters as the states of a Markov Model (MM). This computation provided the probabilities of transition between the different states, estimated as

$$p(s_i, s_j) = \frac{c(s_i, s_j)}{\sum_k \sum_l c(s_k, s_l)} \tag{2.31}$$

where $c(s_i, s_j)$ represents the count of the transitions from state i to state j. It is worth mentioning that $p(s_i, s_j)$ is not a joint probability since i and j are in order. The transition probabilities are defined in such a way that the proportions of each state is reflected.

As can be seen in Figure 2.11, some states have only a single probability of transition (besides the return to the same state) while others have transitions to more than one additional state. Based on their probabilities of transition the states can be connected in a way that reflects possible pathways from U (state 0) to F (state 3). In this scheme

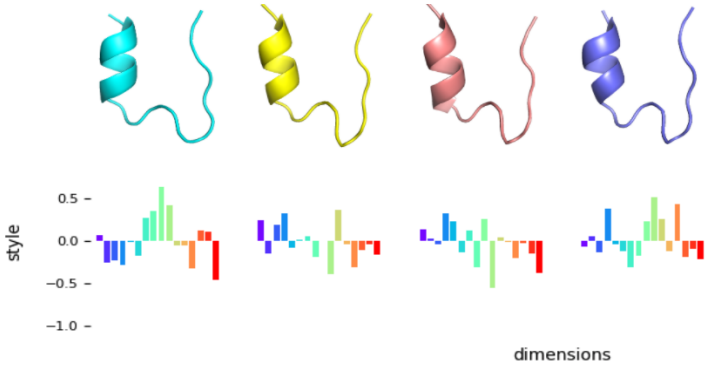

Figure 2.9: Examples of fake frames generated from the AAE-GS model. The corresponding Gaussian noise vectors are plotted as bar plots. All 16 elements of the style representation are spread out along the horizontal axis of each bar plot. Different dimensions are plotted in different colors.

(Figure 2.11), state 2 is a required intermediate in the folding process. In this state, the helix is starting to form. The two prolines (Pro12 and Pro18) are pointing towards each other and, with Tyr3 they are starting to form the "cage". The tryptophan (Trp6) is still outside the cage but as the helix will continue to grow past residue 5, the tryptophan will start to become the center of the cage. It can reach directly the folded state but it can also transition to states 1, 4 and 7. State 1 is a dead-end that has to return to state 2 to reach the folded state 3. State 7 can reach the folded state through two paths: returning to state 2 or transition to state 4 that in turn can reach the folded state 3. State 4 has the largest number of significant transition probabilities: to states 2, 7, 5 and to the folded state (3). It is the last step before reaching the final fold. It has most residues in the correct conformation and would require only an ˜100° counter-clockwise rotation of the helix and small rearrangement to reach the folded state. State 5 is another dead-end. It apprears that in state 5, the protein is attempting to fold as an antiparallel hairpin. Since this arrangement does not lead to favorable interactions nor can it evolve into the final fold. It has to return to state 4 to reach the folded state. State 6 is interesting in the sense that it is a not fully-folded conformation that is only accessible from the folded state and may represent a partially unfolded state in equilibrium with the folded state in the conditions of the simulation. This state 4 has a high probability of transition to itself, second only to that of the folded state. The structural descriptions for states mentions above are reflected in Figure 2.10.

The combination of AAE-GS-identified clusters with a MM using the clusters as the model states, represents an objective analysis of an MD trajectory that provides a highly interpretable model of the folding pathway described by the simulation. The same combination of MD simulations, AAE-GS clustering and MM can be used to identify transitions in enzymes, transporters, channels and others proteins and can become an unbiased procedure to gain insight about the mechanism these proteins.

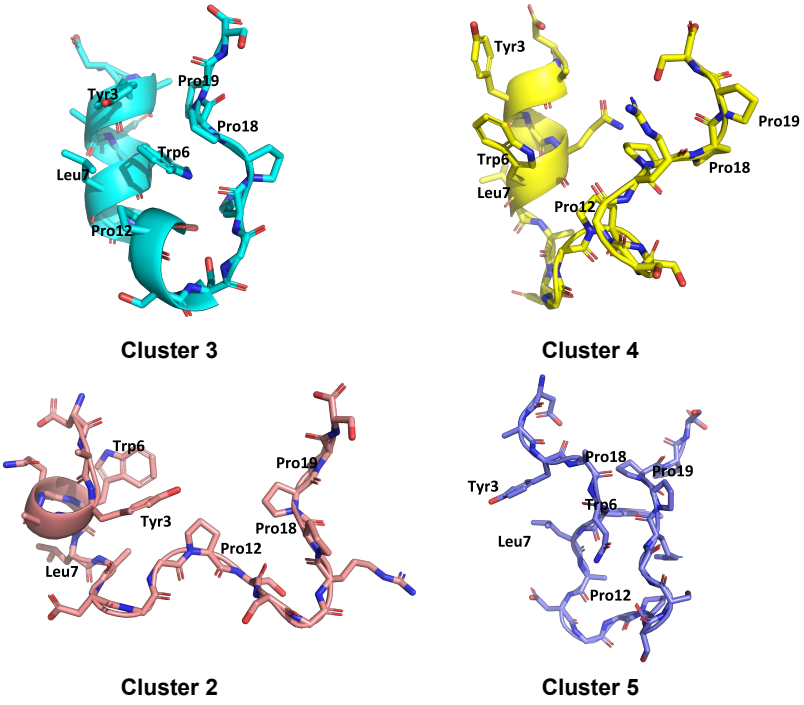

Figure 2.10: Structural descriptions for selective states. The backbone of the protein is plotted with the representation Cartoon and key residues are shown with balls and sticks.

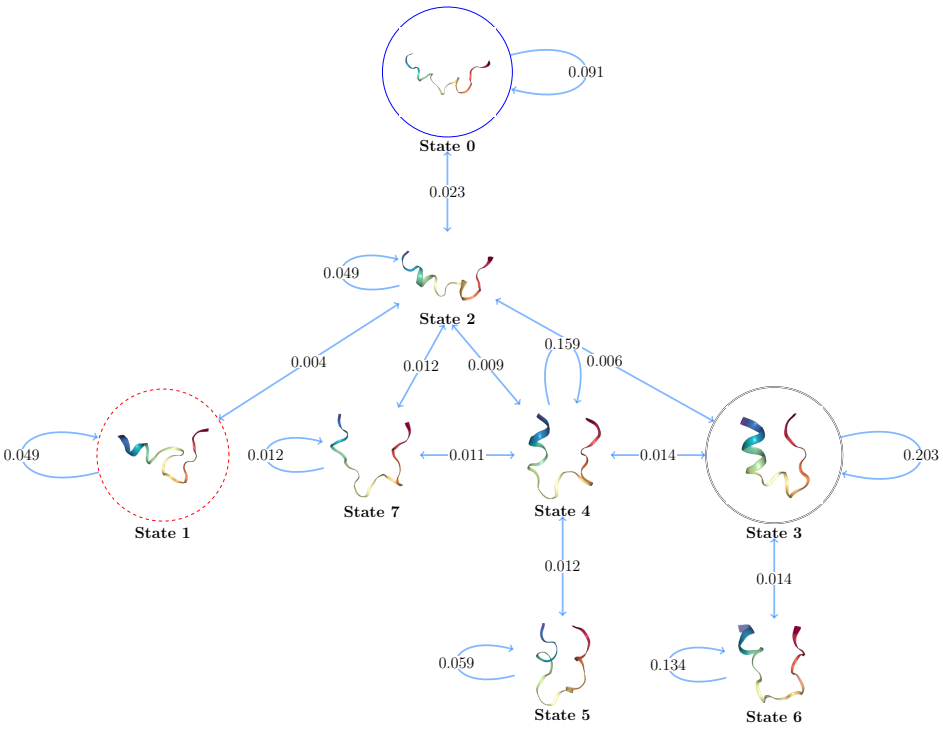

Figure 2.11: Transitions between different states of Trp-cage. The first frame in each cluster is used as the representation of the cluster. The unfolded state is circled in blue. The perfect folded state is double circled in black. A misfolded state (State 1) is circled in red dashes. All transition probabilities are sit in the middle of the arrow. Arrows with a transition probability lower than 0.005 are pruned.

2.3.5 Discussion and Conclusion

Unlike clustering images, clustering MD frames is intrinscally hard since the frame set, which usually comes from trajectories of MD simulation, contains all intermediate conformations of a continuous transformation from one featured state of the macromolecule to another. Clustering such continuous paths is equivalent to determining boundaries at some points of the paths. Consequently, assigning the frames within some neighborhood of the boundaries is somewhat arbitrary. In more formal terms, as our decoder is actually a continuous function, frames with smaller $|\mathbf{z}|$ reveal the specific features of a cluster better than those with larger $|\mathbf{z}|$. For those frames with large $|\mathbf{z}|$, the algorithm is less accurate about their assignment. This indetermination could be resolved by having more clusters, however, more clusters will make features attributed to each cluster less distinguishable. Figure 2.12 and 2.13 plot a random selection of frames from the first 10,000 frames of each cluster with the same clustering results from the AAE and AAE-GS experiments, respectively. We found that the majority of frames in the random selection still preserve specific features that identified their clusters.

From a technical perspective, the general architecture we propose for clustering could be enhanced by involving CNNs and other types of networks. The hyperparameters of all network components are subject to automatic tuning in order to achieve the smallest reconstruction error. Currently, a major caveat of training our model is the lack of robustness of the two GAN components. However, as more and more efforts[19, 20] on stablizing GAN are brought into play in the machine learning community, we expect to have a stable way of training GAN components and tuning the hyperparameters of the discriminators.

The AAE-GS algorithm we propose showed to be effective in clustering MD clusters and it naturally provides a quantity $|\mathbf{z}|$ that represents the loyalty of a frame to the cluster it belongs to. In addition, we show that our clustering results can be useful on revealing and quantifying important biophysical properties of macromolecules.

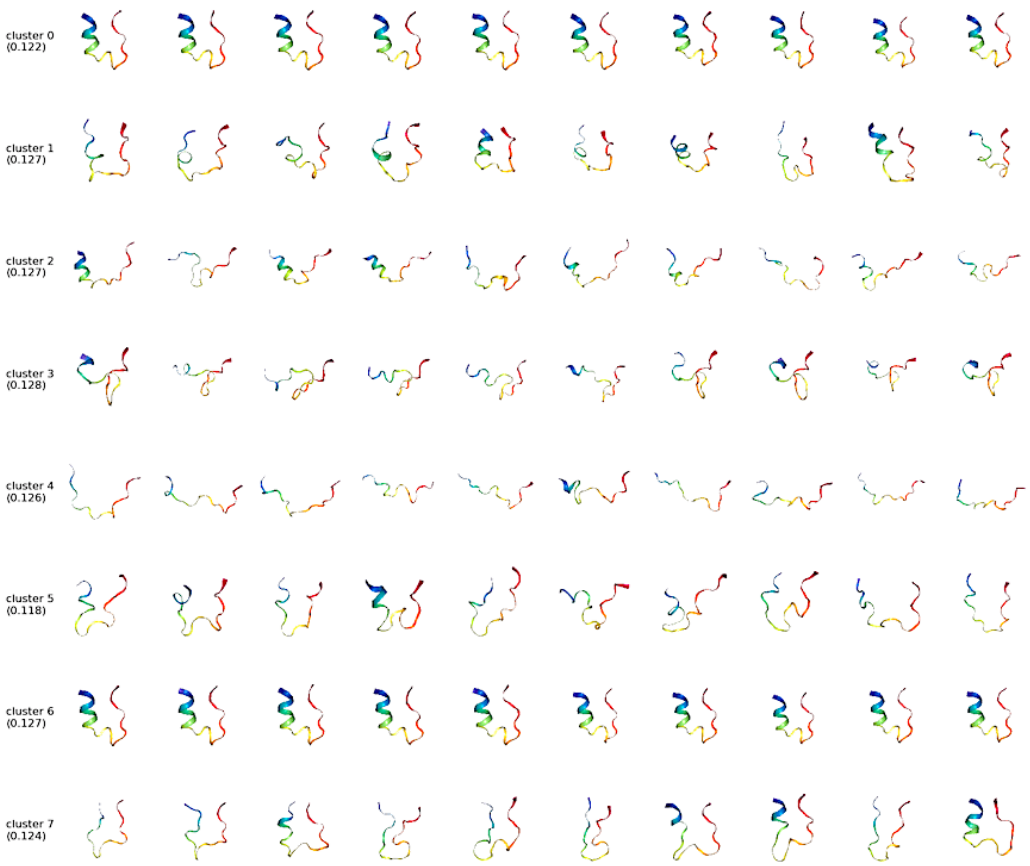

Figure 2.12: Random frames selected from the first 10,000 of each cluster (AAE)

Figure 2.13: Random frames selected from the first 10,000 of each cluster (AAE-GS)

References

[1] Matthew P Harrigan and Vijay S Pande. "Landmark Kernel tICA For Conformational Dynamics". In: *bioRxiv* (2017), p. 123752.

[2] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kale, and K. Schulten. "Scalable molecular dynamics with NAMD". In: *J Comput Chem* 26.16 (2005), pp. 1781–1802.

[3] Mark James Abraham, Teemu Murtola, Roland Schulz, Szilrd Pll, Jeremy C. Smith, Berk Hess, and Erik Lindahl. "GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers". In: *SoftwareX* 12 (2015), pp. 19 –25.

[4] R. Salomon-Ferrer, D.A. Case, and R.C. Walker. "An overview of the Amber biomolecular simulation package". In: *WIREs Comput. Mol. Sci.* 3 (2013), pp. 198–210.

[5] B.R. Brooks, III C.L. Brooks, and Jr. A.D. MacKerell. "CHARMM: The Biomolecular Simulation Program". In: *J Comput Chem.* 30.10 (2009), pp. 1545–1614.

[6] D. E. Shaw, J. P. Grossman, J. A. Bank, B. Batson, J. A. Butts, J. C. Chao, M. M. Deneroff, R. O. Dror, A. Even, C. H. Fenton, A. Forte, J. Gagliardo, G. Gill, B. Greskamp, C. R. Ho, D. J. Ierardi, L. Iserovich, J. S. Kuskin, R. H. Larson, T. Layman, L. S. Lee, A. K. Lerer, C. Li, D. Killebrew, K. M. Mackenzie, S. Y. H. Mok, M. A. Moraes, R. Mueller, L. J. Nociolo, J. L. Peticolas, T. Quan, D. Ramot, J. K. Salmon, D. P. Scarpazza, U. B. Schafer, N. Siddique, C. W. Snyder, J. Spengler, P. T. P. Tang, M. Theobald, H. Toma, B. Towles, B. Vitale, S. C. Wang, and C. Young. "Anton 2: Raising the Bar for Performance and Programmability in a Special-Purpose Molecular Dynamics Supercomputer". In: (2014), pp. 41–53. ISSN: 2167-4329. DOI: 10.1109/SC.2014.9.

[7] S. M. Larson, C. D. Snow, M. Shirts, and V. S. Pande. "Folding@Home and Genome@Home: Using distributed computing to tackle previously intractable problems in computational biology". In: *ArXiv e-prints* (2009). arXiv: 0901.0866 [physics.bio-ph].

[8] Robert T. McGibbon, Kyle A. Beauchamp, Matthew P. Harrigan, Christoph Klein, Jason M. Swails, Carlos X. Hernández, Christian R. Schwantes, Lee-Ping Wang, Thomas J. Lane, and Vijay S. Pande. "MDTraj: A Modern Open Library for the Analysis of Molecular Dynamics Trajectories". In: *Biophysical Journal* 109.8 (2015), pp. 1528 –1532. DOI: 10.1016/j.bpj.2015.08.015.

[9] Naveen Michaud-Agrawal, Elizabeth J. Denning, Thomas B. Woolf, and Oliver Beckstein. "MDAnalysis: A toolkit for the analysis of molecular dynamics simulations". In: *Journal of Computational Chemistry* 32.10 (2011), pp. 2319–2327. DOI: 10.1002/jcc.21787. URL: https://doi.org/10.1002/jcc.21787.

[10] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. "On the Surprising Behavior of Distance Metrics in High Dimensional Space". In: *Lecture Notes in Computer Science*. Springer, 2001, pp. 420–434.

[11] Arthur Zimek, Erich Schubert, and Hans-Peter Kriegel. "A survey on unsupervised outlier detection in high-dimensional numerical data". In: *Statistical Analysis and Data Mining* 5.5 (2012), pp. 363–387. DOI: 10.1002/sam.11161. URL: https://doi.org/10.1002/sam.11161.

[12] Xavier Daura, Karl Gademann, Bernhard Jaun, Dieter Seebach, Wilfred F. van Gunsteren, and Alan E. Mark. "Peptide Folding: When Simulation Meets Experiment." In: *Angewandte Chemie* DOI: 10.1002/(SICI)1521-3773(19990115)38:1/2¡236::AID-ANIE236¿3.0.CO;2-M (1999).

[13] S. Lloyd. "Least Squares Quantization in PCM". In: *IEEE Trans. Inf. Theor.* 28.2 (2006), pp. 129–137. ISSN: 0018-9448. DOI: 10.1109/TIT.1982.1056489. URL: http://dx.doi.org/10.1109/TIT.1982.1056489.

[14] David Arthur and Sergei Vassilvitskii. "K-means++: The Advantages of Careful Seeding". In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '07. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. ISBN: 978-0-898716-24-5. URL: http://dl.acm.org/citation.cfm?id=1283383.1283494.

[15] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN: 0387310738.

[16] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN: 0262018020, 9780262018029.

[17] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. "Adversarial Autoencoders". In: *International Conference on Learning Representations*. 2016. URL: http://arxiv.org/abs/1511.05644.

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[19] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, 2017, pp. 214–223. URL: http://proceedings.mlr.press/v70/arjovsky17a.html.

[20] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. "Improved Training of Wasserstein GANs". In: *CoRR* abs/1704.00028 (2017). arXiv: 1704.00028. URL: http://arxiv.org/abs/1704.00028.

[21] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables". In: *CoRR* abs/1611.00712 (2016). arXiv: 1611.00712. URL: http://arxiv.org/abs/1611.00712.

[22] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical Reparameterization with Gumbel-Softmax". In: 2017. URL: https://arxiv.org/abs/1611.01144.

[23] E.J. Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*. Applied mathematics series. U. S. Govt. Print. Office, 1954. URL: https://books.google.com/books?id=SNpJAAAAMAAJ.

[24] Chris J. Maddison, Daniel Tarlow, and Tom Minka. "A* Sampling". In: *Advances in Neural Information Processing Systems 27*. 2014.

[25] K. Lindorff-Larsen, S. Piana, R.O. Dror, and D.E. Shaw. "How fast-folding proteins fold." In: *Science* Oct 28;334(6055):517-20. (2011).

[26] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. "TensorFlow: A system for large-scale machine learning". In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 2016, pp. 265–283. URL: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

[27] Matthew P Harrigan, Mohammad M Sultan, Carlos X Hernández, Brooke E Husic, Peter Eastman, Christian R Schwantes, Kyle A Beauchamp, Robert T McGibbon, and Vijay S Pande. "MSMBuilder: statistical models for biomolecular dynamics". In: *Biophysical journal* 112.1 (2017), pp. 10–15.

[28] Christopher D Snow, Bojan Zagrovic, and Vijay S Pande. "The Trp cage: folding kinetics and unfolded state topology via molecular dynamics simulations". In: *Journal of the American Chemical Society* 124.49 (2002), pp. 14548–14549.

[29] Fabrizio Marinelli, Fabio Pietrucci, Alessandro Laio, and Stefano Piana. "A kinetic model of trp-cage folding from multiple biased molecular dynamics simulations". In: *PLoS computational biology* 5.8 (2009), e1000452.

[30] Jed W Pitera and William Swope. "Understanding folding and design: Replica-exchange simulations of"Trp-cage"miniproteins". In: *Proceedings of the National Academy of Sciences* 100.13 (2003), pp. 7587–7592.

[31] Ruhong Zhou. "Trp-cage: folding free energy landscape in explicit water". In: *Proceedings of the National Academy of Sciences* 100.23 (2003), pp. 13280–13285.

[32] J Juraszek and PG Bolhuis. "Sampling the multiple folding mechanisms of Trp-cage in explicit solvent". In: *Proceedings of the National Academy of Sciences* 103.43 (2006), pp. 15859–15864.

Chapter 3

Ion Transport Mechanism of NIS

3.1 Introduction

The Na^+/I^- symporter (NIS), a member of the solute carrier family (SLC5), is the key plasma membrane protein that mediates active I^- uptake in the thyroid gland and other tissues [1], and the first step in the biosynthesis of the thyroid hormones (THs), of which iodine is an essential constituent[2]. NIS-mediated radioiodide treatment has been used in clinical practice to treat thyroid cancer for more than 60 years as a successful targeted internal radiation anticancer therapy [3–6]. The extensive characterization of NIS at the molecular level began with the 1996 isolation of the cDNA that encodes NIS [7]. [8] However, despite efforts [9, 10] made previously and the physiological and biomedical importance of this molecule, the mechanism of I^- transport by NIS remain uncovered. In this chapter, we describe our investigations on the ion transport mechanism of NIS with computational approaches.

NIS couples the "uphill" inward transport of I^- against its electrochemical gradient to the "downhill" inward translocation of Na^+ down its electrochemical gradient, which is generated by the Na^+/K^+ATPase [2, 11]. I^- transport mediated by NIS is electrogenic, with a 2 Na^+:1 I^- stoichiometry [12]. Based on our previous experimental data and statistical thermodynamics (ST) analysis, we proposed a mechanism for the active transport of I^- by NIS under physiological conditions (See Figure 3.1). This mechanism can be summarized into two points: 1) different conformations of the protein expose ligand binding sites to either extracellular or cytoplasmic side, favoring ions uptake or

release, respectively; 2) two Na^+ binding sites, named Na1 and Na2, interact allosterically with the I^- site. 1) can be extrapolated from a well-accepted hypothesis that specific protein functions always associate with particular conformational states, while 2) was supported by thermodynamics data presented in our previous studies [9, 10].

Validating the proposed mechanism experimentally remains impossible because so far, no experimental technique can access the conformational dynamics of NIS at atomic resolution. Due to this fact, computational approaches such as molecular dynamics (MD) simulations and other analytical methods are employed to sample conformational states of NIS, to describe state transitions quantitatively and to link conformations with ion binding or release. In our previous research [10], a NIS homology model was built based on the structure in the inwardly facing conformation of another member of the SLC5 family, the Na^+/galactose cotransporter of *Vibrio parahaemolyticus* (vSGLT). NIS and vSGLT share 32% sequence identity and 64% similarity (between core residues 50 and 456) and thus aligning the homology model with vSGLT (PDB code: 2XQ2) yields a root-mean-square deviation (RMSD) of 1.1 Å. Moreover, inspired by the fold of the *Aquifex aeolicus* Na^+-dependent leucine transporter (LeuT) and the shared structural characteristics of other Na^+-driven transporters, we previously proposed the existence of the Na2 binding site of NIS, identified it via both short MD simulations and verified experimentally the Na2 site. In addition, the Na1 and I^- binding sites were proposed but not validated experimentally. Besides identifying binding sites, no explicit solvent MD experiments were attempted to understand the conformational transitions, their onsets and the binding or unbinding paths of ions.

To extend the scope of the previous research and tackle the remaining questions, we refined the explicit-solvent simulation system used previously [10] by enforcing ion gradients and used the new system to perform multiple long-trajectory MD simulations to sample conformations of NIS as well as to identify paths of ion binding and release. In our 1 μs long trajectory starting with no bound ions, we didn't observe any ion binding paths nor any salient inward-to-outward conformation change. This fact is due to the potential high energy barrier between conformational states of NIS, which is consistent

with the experimental kinetic data [10]. To enhance the probability of observing conformational changes, we placed ions in each of the proposed or hypothesized binding sites, enforced position restraints on those ligands and ran a series of MD simulations maintaining the restraints. Salient conformational changes were observed in some of the restraint simulations. To identify states explored in all our simulations, we employed the AAE-GS clustering algorithm we developed to cluster all the frames we collected in all trajectories. We then picked the restraint trajectories with appropriate conformations and extended them with the position restraints removed. The extended trajectories helped us identify key residues responsible for ion transport and partially reveal their paths. Our results provide essential implications in designing experiments to further validate the ion transport path of NIS and quantitative insights into how conformations of NIS associate with its functions.

3.2 Methods

3.2.1 MD Simulations

3.2.1.1 The Construction of NIS Simulation System

We oriented the homology model of NIS by aligning it with one chain of the vSGLT model presented in the Orientations of Proteins in Membranes (OPM) database. The oriented NIS model was embedded on a POPC lipid bilayer with 110 POPC molecules per leaflet. The membrane-protein system was solvated by two 25 Åthick TIP3P water layers on each side of the membrane. To mimic the physiological electrochemical gradient across the membrane, different numbers of four types of ions (Na^+, K^+, Cl^-, I^-) were added on both sides of the membrane by randomly replacing same number of TIP3P water molecules. The total number of cations added yields a salt concentration of 0.15 M on both side of the membrane. On the extracellular side, the ratio between the number of Na^+ and K^+ is 5:1, while on the intracellular side, this ratio is 1:5. A certain number of Cl^- ions were initially added to neutralize the whole system. Then 12 of them were randomly selected and replaced by iodide ions, which results in 2 I^- ions locating at the extracellular side and 10 at the intracellular side.

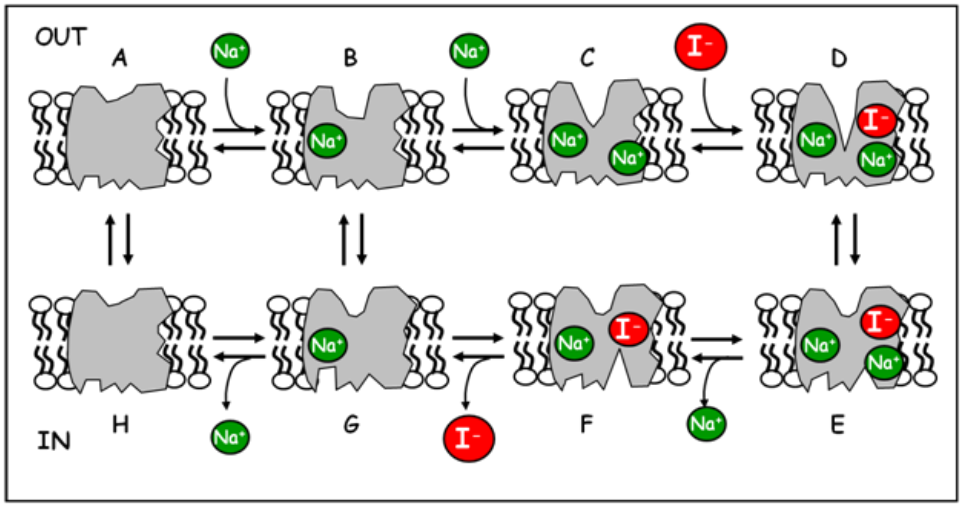

Figure 3.1: Previously proposed working mechanism of NIS. A–H represents different conformational states of NIS, which favor different ion binding behaviors. This mechanism assumes the order of ion bindings and ion releases. No ion translocation inside the tunnel of the transporter is assumed. [13]

Since periodic boundary conditions (PBC) are used during all the NIS simulations, the ions we put on both extracellular and intracellular sides could move accross the periodic boundary, which would alter the electrochemical gradient. To keep ions always on one side of the cell, another DLPC lipid bilayer with a total number of 292 DLPC molecules was added to the bottom of the water box to maintain the ion concentrations at each side of the NIS membrane by isolating the two sides. DLPCs were chosen as the component of isolation layer because they contain fewer atoms, which helps minimize the size of the whole simulation system. After the above construction procedures, the explicit-solvent simulation system of NIS contains 125239 atoms and forms a waterbox with an approximate size of $97 \times 97 \times 147$ (Å), as shown in Figure 3.2.

To target different aims, several simulation experiments of NIS were conducted using different conditions. In some experiments, we treated the first two Na^+ ions among all Na^+ ions listed in the PDB file and the first I^- ion as the ligands of NIS and relocated some or all of them to their proposed binding sites [10] by manually modifying their coordinates. All simulation systems constructed with the above procedures, though with minor differences, underwent the same procedure of energy minimization and equilibration. Energy minimization, equilibration and the following production MD simulations were carried out with GROMACS (version > 2016) [14] on MARCC's GPU nodes. We used the CHARMM36 force field [15] for all NIS simulations.

3.2.1.2 Energy Minimization and Equilibration

Energy minimization was carried out with the steepest descent algorithm. The atoms of the protein and the lipids were fixed during this process by enforcing very hard position restraints. (Section 3.2.1.4) For example, $k_{backbone} = 4000kJ/nm^2$ and $k_{sidechain} = 2000kJ/nm^2$ are used for the protein. The system is considered as energy minimized after either of the following two conditions are satisfied: 1) the maximum force becomes less than 1000. 2) 5000 steps are reached. In most of our cases, the first condition was satisfied.

We equilibrated the simulation system in 6 subsequent runs. These 6 runs "thaw"

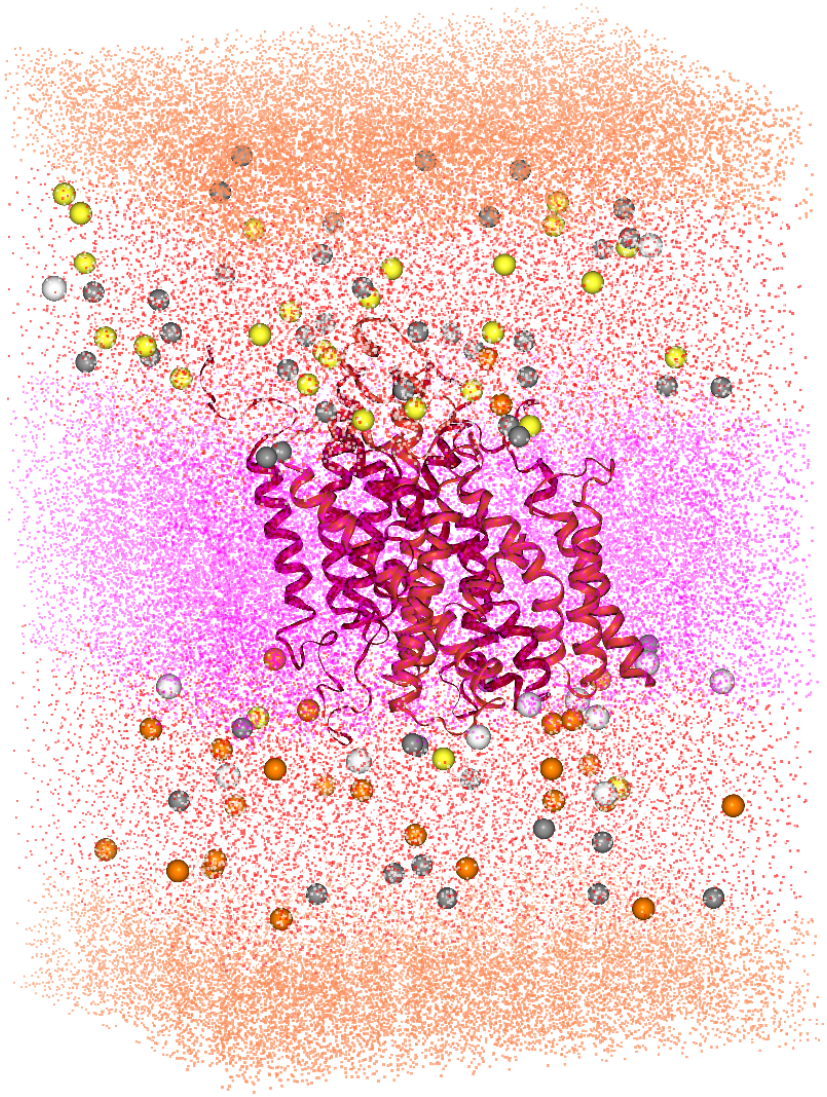

Figure 3.2: NIS simulation system. The protein in the middle of the system is plotted in the cartoon representation. POPC lipid bilayer is shown in purple points and DLPC lipid layer in orange points. All water molecules are presented in red points. Ions are generally plotted with the hyperball representation. Na^+, K^+, Cl^-, I^- ions are colored with yellow, orange, gray and white, respectively.

Step ID	Length	dt
1	25	1
2	25	1
3	25	1
4	100	2
5	1000	2
6	1000	2

Table 3.1: Other parameters used in equilibration runs. k in unit kJ/nm^2 and lengths in unit picosecond (ps). "dt" represents the step size, which is in unit femtosecond (fs).

the protein and the lipids gradually by decreasing the position restraint constants. The first two runs are considered as NVT equilibration, in which Berendsen temperature coupling were used. The rest 4 runs are NPT equilibration, in which Berendsen algorithm was applied on both temperature and pressure coupling. LINCS constraint algorithm was used in energy minimization and equilibration as well as production runs. The temperature of the system was set to 310.15 K. The length and the step size of each run are listed in Table 3.1.

The simulation systems and GROMACS configuration files for energy minimization and equilibration were partially prepared with CHARMM-GUI [16].

3.2.1.3 Non-Restraint Production Simulations

Production simulations were carried out with the equilibrated systems. No position restraint was enforced on any atom of the system unless specified. In all production runs, we adopted 2 fs as the step length and frames were saved to trajectories every 10000 steps (20 ps). We maintained the temperature of system at 310.15 K by applying Nose-Hoover thermostat and the pressure of the system at 1.0 bar with Parrinello-Rahman pressure coupling algorithms. We set the cutoff disance for Lennard-Jones interactions and Coulomb electrostatics interactions to be 12 Å. Particle-Mesh-Ewald (PME) method was used to calculate the long range electrostatics interactions.

We attempted 5 non-restraint MD simulations with the following starting states: unbound (empty), Na^+ bound on Na1 site (Na1), Na^+ bound on Na2 site (Na2), both Na^+ occupied (Na1Na2) and fully bound with all three ions (full). Observations on all partially bound and fully bound simulations showed that ions left their binding sites

after equilibration or after first 2 ns of the production simulations. Unfortunately, this result contradicts with our previous results [10]. It is not clear that the contradictions are caused by the ion gradients we imposed in the new simulation system. Because of no ions binding observed in those non-restraint trajectories, we abandoned all simulations except the empty one. The empty trajectory was extended to 1 μs.

3.2.1.4 Restraint Simulations

Another series of MD simulations were carried out with position restraints to find the onset of outward-open conformations, which might be induced by enforcing ion binding.

Applying position restraints to atoms aims at restraining them to fixed reference positions, denoted by $\boldsymbol{R}$. It is implemented by adding extra harmonic terms V_{pr} to the potential function of the simulation system. Mathematically, V_{pr} is defined as

$$V_{pr}(\boldsymbol{r_i}) = \frac{1}{2} k_{pr} |\boldsymbol{r_i} - \boldsymbol{R_i}|^2 \tag{3.1}$$

where $\boldsymbol{r_i}$ denotes the position vector of atom i and k_{pr} represents the position restraint constant. V_{pr} applied to a specific atom artificially creates a energy barrier for it to move. This approach is widely used in the energy minimization and equilibration. In our experiments, we applied position restraints to different combinations of ligands and the restraints are applied starting at the equilibration phase.

Non-restraint extensions of some restraint simulations were run to validate the stability of the restraint-induced conformation and to check its functionality. We run a series of 1 ns simulations to decay the position restraints k_{pr} to zero with a gradient of 800, 500, 200, 100, 50, 20. Then extended simulations were run as conventional MD production simulations. The length of each extension is 200 ns. For the convenience of description, we summarize and label all the trajectories in Table 3.2.

Label	Description	k_{pr}	length
empty	No ions placed in binding sites	none	1 μs
Na1-fixed	Restraining Na^+ to Na1 site	1000 (2.39)	200ns
Na2-fixed	Restraining Na^+ to Na2 site	1000	200ns
Na1Na2-fixed	Restraining Na^+ to Na1 and Na2 site	1000	125 ns
all-fixed-200	Weakly restraining all ions on z-dimension	200 (0.48)	1 μs
all-fixed-400	Weakly restraining all ions on z-dimension	400 (0.96)	550 ns
all-fixed-1000	Restraining all ions on z-dimension	1000 (2.39)	130 ns
Na2-fixed-ext	Extending Na2-fixed with no restraints	none	200 ns
Na1Na2-fixed-ext	Extending Na1Na2-fixed with no restraints	none	200 ns

Table 3.2: NIS trajectories. k_{pr} in unit kJ/nm^2 ($kcal/\text{Å}^2$)

3.2.2 AAE-GS Clustering

To identify the states explored in all trajectories and find paths linking some of the states identified, we did an unbiased clustering of all the frames we collected in all the trajectories using the AAE-GS algorithm we developed. All the GROMACS XTC trajectories were converted to shards of Tensorflow record files (TFRecord) by aligning all trajectories to a standard NIS structure, extracting only the C-α coordinates and flattening them into the TFRecord format. All shards were shuffled and read into the program in minibatches size of 1024.

The AAE-GS architecture and the hyperparameters we used for clustering NIS trajectores are listed in the following table (Table 3.3).

As Table 3.3 shows, we clustered frames into 25 clusters and we set the dimension of the style representation as 4. We assumed that the style representation follows independent identically distributed (i.i.d) standard Gaussians. The learning rate for the autoencoder component is set to 0.00002 and the rates for both GAN compoents are set to 0.0001. We used 10 as the gradient penalty weight for regularizing the weights of the discriminators. The temperature parameter τ of the Gumbel-Softmax distribution was annealed from 0.1 to 0.01 in 50000 steps by using a polynomial decay of power 4. All components were trained with the Adams optimizer for more than 100,000 steps. In each step, the autoencoder and the discriminators of GANs were updated 5 times and the generators updated only once.

Component	Architecture[1]
Shared Encoder	$2000_{\text{BN, 0.1 DO}} \times 2000_{\text{BN}} \times 400_{\text{BN}}$[2]
Categorical Encoder	$100_{\text{BN}} \times 25^{\text{softmax}}$
Style Encoder	$200_{\text{BN}} \times 4^{\text{id}}$
Categorical Decoder	$100_{\text{BN, 0.1 DO}} \times 400_{\text{BN}}$
Style Decoder	$200_{\text{BN, 0.1 DO}} \times 400_{\text{BN}}$
Shared Decoder	$2000_{\text{BN}} \times 2000_{\text{BN}} \times 1479^{\text{id}}$
Discriminator (Cat-GAN)	$400_{\text{0.1 DO}} \times 400 \times 400 \times 200_{\text{0.1 DO}} \times 200 \times 200 \times 200 \times 100 \times 1^{\text{id}}$
Discriminator (Style-GAN)	$400_{\text{0.1 DO}} \times 400 \times 400 \times 200_{\text{0.1 DO}} \times 200 \times 200 \times 200 \times 100 \times 1^{\text{id}}$

Table 3.3: Structures of all sub-components of our AAE-GS model.

[1] All components are modeled with multilayer perceptrons (MLP); we represent the layers of MLP with numbers separate with "×". Each number refers to the number of nodes in that layer.

[2] Note for the subscript and superscript: "BN" refers to Batch Normalization; "DO" prefixed with a number refers to dropout and the number is the dropout probability; Superscript refers to an activation function other than "relu". We use "relu" as the default activation function.

3.3 Results

3.3.1 Restraint-Induced Conformation Change

In the 1 μs empty trajectory, no ion binding events were detected and salient conformational changes were hardly observed. To search for conformations that allow ions binding, we conducted a series of MD experiments with multiple levels of position restraints (See Table 3.2) applied to different combinations of ion ligands. To accommodate the fixed ions, NIS was likely to adjust its conformation quickly so that we can observe the change of conformation without running an extensively long simulation. Naive visualizations on the position restraint trajectories found that the trajectory Na2-fixed, Na1Na2-fixed and all-fixed-1000 showed salient conformational changes. To validate the visualized conformational changes, we selectively extended two trajectories (Na2-fixed, Na1Na2-fixed) with the restraints removed, which allow testing the robustness of the induced conformations, observing its interactions with ion ligands and its transitions to the other possible conformations.

To seek an accurate classification of all 177,741 frames collected in all the above trajectories, we analyzed them with the AAE-GS clustering algorithm we developed. We intended to cluster all frames into 25 clusters, however, the number of clusters turned out to be 20 after 100,000 steps of training. The autoencoder's loss converged to $\sim$0.01 nm^2, which indicates that the resolution of the clustering is $\sim$1 Å and the trained style representations of all the frames shaped as Gaussian (Figure 3.3). A distribution of frames grouped by cluster ID and the trajectory they were sampled from is listed in Table 3.4.

The AAE-GS clustering algorithm clustered most of the frames in a trajectory into one or two dominating clusters. For example, frames in the empty trajectory were grouped into cluster 2 and 6 and frames in trajectory Na2-fixed were classified into cluster 15 and 24. Overlap of the frame distribution between one trajectory and another is scarce. This is because position restraints limit the exploration in the conformational space of MD to some particular local domains. In other words, conformations that are

not in favor of the restraints will not be sampled and shown in the resulting trajectories. The clustering algorithm diligently distinguished shuffled frames from different trajectories and relabeled them nicely. In addition, it is remarkable that frames from the trajectory Na2-fixed-ext were all classified into the dominating cluster of Na2-fixed, its origin and so were the frames from Na1Na2-fixed-ext. Such evidence implies that the AAE-GS algorithm is quite effective in recognizing the subtle distinctions between each two trajectories and it also credits the position restraints for inducing conformations with different characteristics.

To understand the characteristics of the induced conformations, we inspected the cluster head of each cluster, the frame whose style representation is closest to the mean of the 4-dimensional Gaussian distribution among all the frames in that cluster and compared pairs of the cluster heads of interest with visualization softwares (PyMOL[17], NGLViewer[18]). We used cluster heads as representatives of each cluster since to some degree, they could be considered as the closest frame to the weighted average structure of each cluster. In that sense, comparing heads of clusters reveals significant conformational differences between the clusters they represent so as the trajectories they came from. We present detail inspections on the heads of cluster 6 (C6), the dominating cluster of the empty trajectory, and of cluster 15 (C15), the dominating cluster of Na2-fixed in Figure 3.4 and 3.5. This pair was chosen based on our naive observations on all the trajectories, where we found that Na2-fixed had a salient inwardly-to-outwardly open conformational change.

To identify the potential outward-open features lying in C15, we aligned both C6 and C15 with a crystal structure of Na^+-coupled sialic acid symporter (SiaT, PDB code: `5NVA`), which also adopts LeuT fold and has an outwardly open conformation [19]. The RMSDs between either C6 or C15 and 5NVA are around 4 to 5 Å, which indicates that NIS and SiaT share a similar protein architecture. Comparing C6 and C15 with SiaT shows the extracellular portions of some transmembrane (TM) helices of C6 are more tilted to the central axis of the transporters (Figure 3.4 A), while their counterparts of C15 align better with SiaT. The orientations of particular TM helices of C15, e.g.,

the TM helix pointed by the purple arrow in Figure 3.4 B, shows a tendency of being more outwardly-open than SiaT. An alignment of C6 and C15 (Figure 3.4 C) clearly shows the features of outwardly-open. Besides structural alignments, we examined the electrostatic surfaces of C6, C15 and SiaT (Figure 3.4 D, E and F). Seen from the extracellular side, C6 has no visible access on the internal cavity, while both C15 and SiaT have limited accessibility. Observed from cytoplasmic side, both C6 and C15 tend to be more inwardly-open than SiaT (Figure 3.5 A, B and C), which can be validated from the their electrostatic surfaces (Figure 3.5 D, E and F). The internal cavities of C6 and C15 are exposed to the cytoplasm while the cavity of SiaT is concealed. In conclusion, we suggest that C6 has an inwardly-open conformation and C15 shows the onset of the transition between the inwardly- and outwardly-open states.

Though the population of different states in each trajectory can be accurately estimated, associations between conformational states and their functions remains hard to reveal. The majority of our simulations were conducted under restraint, which yields trajectories containing homogenous frames. Estimating the transition probabilities between conformational states needs sufficient, non-restraint sampling and so does calculating emission probabilities[3] that describe the binding between the ions binding behaviors and each conformational state. No MD simulations are attempted to resolve this aim in this chapter.

3.3.2 Validation of Ion Binding Sites

In both of Na2-fixed-ext and Na1Na2-fixed-ext simulations, ion transport and binding behaviors are observed, which validates the previously proposed binding sites. By visualizing both trajectories, we found abundant ion binding information in Na1Na2-fixed-ext but very few (< 5 ns) in the other. In Na1Na2-fixed-ext, a Na^+ ion significantly bound to Na2 site for ~ 170 ns, while occupancies of other binding pockets lasted shorter times. To identify key residues that form potential binding pockets, we excerpted a 25 ns trajectory that contains binding activities of all three ion ligands. For each ion ligand, we selected residues in contact with it (within 5 Å) in each frame and merged them into

[3]In the sense of hidden Markov model (HMM).

Cluster ID	Trajectory	Number of Frames
0	empty	43
2	empty	21853
5	empty	3182
6	empty	24869
15	empty	10
18	empty	33
23	empty	11
0	Na2-fixed	1253
5	Na2-fixed	6
6	Na2-fixed	10
15	Na2-fixed	5202
18	Na2-fixed	16
24	Na2-fixed	3514
10	Na1Na2-fixed	6150
15	Na1Na2-fixed	48
22	Na1Na2-fixed	35
11	Na1-fixed	9744
15	Na1-fixed	254
22	Na1-fixed	3
5	all-fixed-1000	163
15	all-fixed-1000	200
18	all-fixed-1000	30
24	all-fixed-1000	3608
0	all-fixed-200	556
5	all-fixed-200	172
10	all-fixed-200	152
14	all-fixed-200	8
15	all-fixed-200	138
16	all-fixed-200	15901
18	all-fixed-200	76
19	all-fixed-200	23995
23	all-fixed-200	1071
24	all-fixed-200	7932
1	all-fixed-400	1009
4	all-fixed-400	6836
5	all-fixed-400	389
7	all-fixed-400	13458
9	all-fixed-400	151
14	all-fixed-400	5135
15	all-fixed-400	35
17	all-fixed-400	2
18	all-fixed-400	6
20	all-fixed-400	474
23	all-fixed-400	6
15	Na2-fixed-ext	10001
10	Na1Na2-fixed-ext	10001

Table 3.4: A distribution of frames grouped by their cluster ID and the trajectories they are sampled from.

separate residue sets. For each residue in the set, we stacked the closest distances, represented as colors from light red (farthest) to dark red (closest), between the ion and the residue into a column ordered by time. Columns showing a less than 2 ns accumulated time of contact are filtered and the rest of columns line up horizontally into a heatmap. The benefit of using such heatmap is that it can reveal time-based contacts between the ligand and the protein. Residues forming binding pockets along the tunnel of NIS as well as the path of ion transport through various binding pockets can be illustrated directly. Figure 3.6, 3.7 and 3.8 present the heatmaps generated for I^-, NaA (the Na^+ bound to Na1) and NaB (the Na^+ bound to Na2), respectively.

Key residues composing the Na2 site found in our experiments are in good agreement with those previously proposed. Our data (Figure 3.6) shows significant interactions between NaB and D191, M68, S353, S66, S69, T354. A frame that best represents the Na2 site was used to illustrate this site (Figure 3.9). Among all residues we identified, D191, S353, S66 are described before while M68 is a newly identified residue showing strong contact signals. In our simulations, substantial translocation of NaB to another binding pocket is not observed. In the end, NaB dissociated from the Na2 site and was released into cytoplasmic side in less than 2 ns.

Translocation of NaA and I^- between different binding pockets were observed. We identified two potential binding pockets for each of NaA and I^- one of which is adjacent to the extracellular pore (upper) and the other lies close to the cytoplasmic pore (lower). For each ion, both binding pockets together with the key residues are illustrated in Figure 3.10 and 3.11 in different views. The upper binding pocket of I^- consists of Q94, Q72, W255 and V254 (Figure 3.10 C). The side chain of M258 points to the extracellular direction to support this binding pocket. The lower binding pocket of I^- consists of the T354, T357, S358, Y144, Y137 and Q263 (Figure 3.10 D). This result partially agrees with what proposed previously (F67, Q72, Q94, M258) and furthermore, shows that I^- could also interact with the lower site. As for two binding pockets for NaA, we found two set of residues which are quite different from what previously reported. The upper binding pocket, which consists of E79, L413 and F417, serves essentially as a gate for

NaA. Figure 3.7 shows that NaA interacts with the upper binding pocket for about 2 ns and then it quickly translocates to the lower site majorly represented as S257, M258 and Q263. Interactions with Y144, L143 and M420 are also detected but with weaker signals.

3.3.3 Transport of NaA and the Iodide Ion

As presented above, in the Na1Na2-fixed-ext trajectory, the transport of NaA and I^- from extracellular side to the lower pockets were observed. The whole transport process happened while NaB was tightly bound to the Na2 site. In this trajectory, I^- came into the upper binding pocket before NaA approached the protein. I^- was stuck in the upper pocket and the side chain of residue M258 points toward the extracellular direction to prohibit the release of I^- (Figure 3.12 A). NaA came into the upper pocket and quickly altered the orientation of M258's side chain, which allows the I^- translocation to the lower pocket (Figure 3.12 B). According to this observation, M258 potentially plays an important role in translocating I^- from the upper binding pocket to the lower. Though cooperative behaviors are found in our trajectory, the total ordering of ion binding still remains unknown. No information on whether NaB binds prior to binding of the other ions are provided in all the trajectories.

The transport paths of both NaA and I^- can be generally described as 1) moving from extracellular side to the upper binding pocket, 2) translocating from the upper binding pocket to the lower and 3) releasing the ions from the inner binding pocket to the cytoplasm. Cooperation between NaA and I^- is observed, which potentially explains the ion coupling mechanism of NIS.

3.4 Conclusions

The goal of our computations was to identify the conformations that are compatible with binding of the ions as well as the onset of the transitions that lead to these conformations and eventually to the full transport cycle. As ion binding events are infrequent, we adopted three strategies in addition to the usual MD computations to enhance the probability of observing these events and to achieve our goal:

1. We created and maintained ion-gradients that were resilient to being depleted by the periodic boundary conditions.

2. We developed and used a AAE-GS clustering algorithm for identifying clusters of relevant conformations (states).

3. We computed trajectories with different combination of ions that were restraint to the positions previously identified as the ions binding sites.

All these strategies together led to an enhanced observation of transport-relevant events. We found that MD simulations with position restraints trigger swift conformational changes of protein. Moreover, an effective clustering method, e.g., AAE-GS, was capable of identifying conformational states with different characteristics based only on the coordinates of the protein backbone. Combining position restraint simulations and clustering, we identified the onset of the inwardly-to-outwardly open conformation (C15), which dominates one of the restraint simulations. A follow-up MD simulation (Na1Na2-fixed-ext) conducted based on this conformation shows that the conformation is robust with no position restraints imposed and it favors the ion binding more than the original inwardly-facing of NIS.

By analyzing the trajectory of Na1Na2-fixed-ext, we identified multiple key residues closely interacting with the ion ligands. The identification of some residues, such as, T354, S66, Q263 and M258 strongly validates the results we obtained in the previous research. Besides, new residues, especially, residues composing the Na1 site (NaA binding pockets) are proposed for the first time, which provides new implications for designing experiments to validate the effects of those residues and insights into their mechanistic roles in ion transport. In addition to identifying static binding sites, a dynamic process of ion transport was partially observed in the trajectory Na1Na2-fixed-ext. NaA and I^- are cooperatively transported from the extracellular side to the lower pockets and we found NaA plays a key role in altering the orientation of a specific residue, M258, which facilitates the translocation of I^-. This coupling mechanism, needs to be justified with more MD simulations and experiments.

Although many questions about the ion transport mechanism remains unanswered, for example, the ordering of the binding and the association between conformational states and ion binding, our computational research provides valuable insights in characterizing inwardly- and outwardly-open conformations and understanding ion binding and translocation at atomic level. More generally, our research can also be considered as a successful example of using position restraint simulations to trigger conformational changes in a short time as well as applying AAE-GS clustering algorithm to automatically compare and classify conformations. The methods use for the analysis of NIS transport are general and can be applied in other systems to address questions involving conformational changes in proteins.

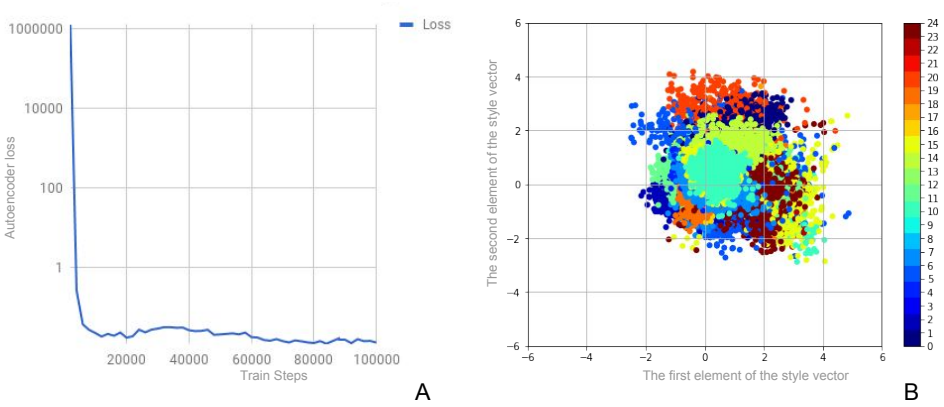

Figure 3.3: Convergence of AAE-GS. The convergence of the autoencoder loss is plotted in A and the first two dimensions of the trained style representations are shown in B.

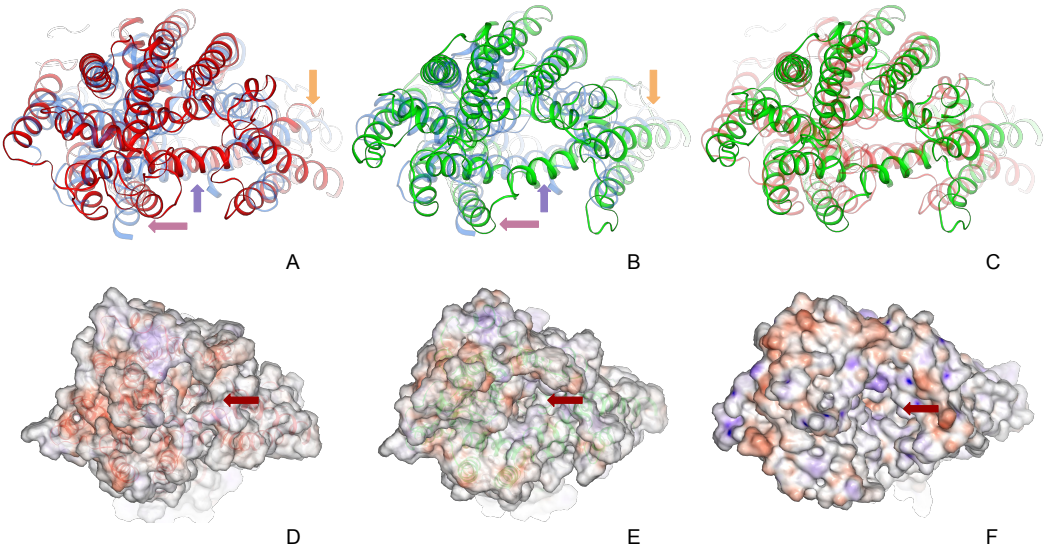

Figure 3.4: The validation of the outward-open conformation from the extracellular perspective. Figure A, B and C represent protein in Cartoon, where SaiT is colored in blue, C6 colored in red and C15 colored in green. A degree of opacity is used for plotting SaiT to avoid unclear overlap between structures. Figure D, E and F plots the electrostatic surfaces of C6, C15 and SaiT, respectively. Arrows with a same color are used to mark a local region subject to structural comparision.

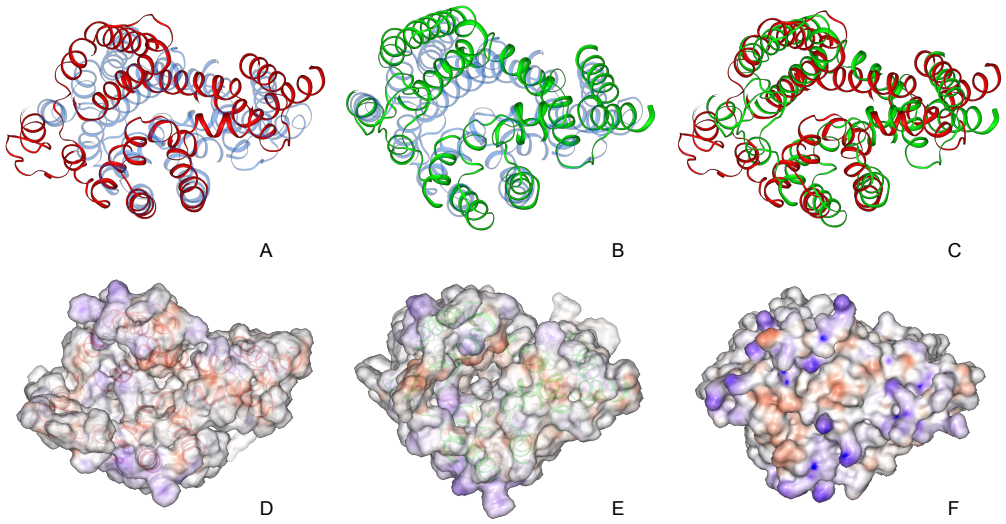

Figure 3.5: The validation of the inward-open conformation from the intracellular perspective. Figure A, B and C represent protein in Cartoon, where SaiT is colored in blue, C6 colored in red and C15 colored in green. A degree of opacity is used for plotting SaiT to avoid unclear overlap between structures. Figure D, E and F plots the electrostatic surfaces of C6, C15 and SaiT, respectively. Arrows with a same color are used to mark a local region subject to structural comparision.

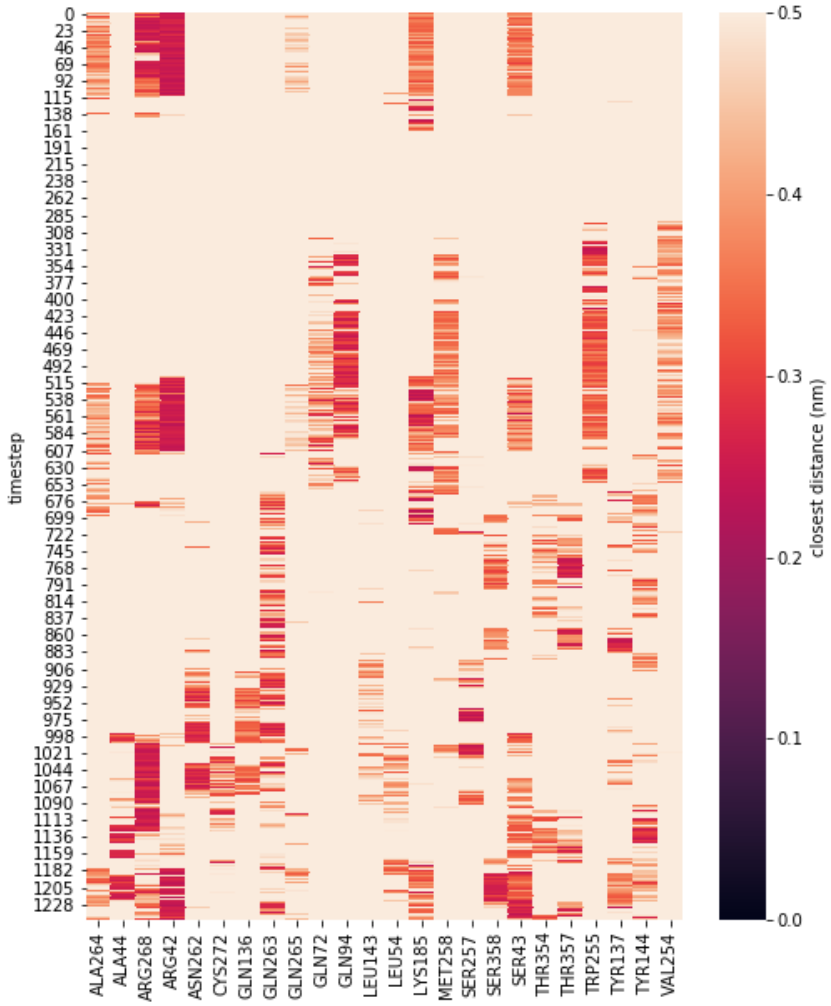

Figure 3.6: Frequent contacts between I^- and residues. Frame indices are plotted on the vertical axis and residues were displayed on the horizontal axis. Residues making accumulated contacts shorter than 2 ns are abandoned.

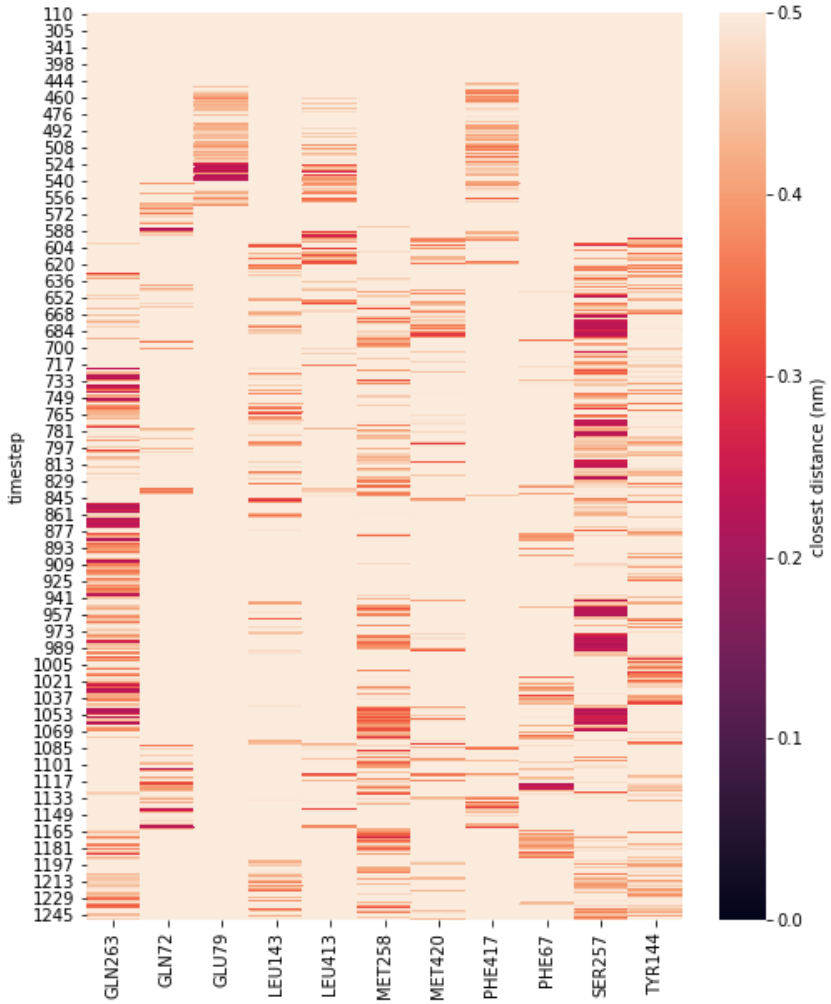

Figure 3.7: Frequent contacts between NaA and residues. Frame indices are plotted on the vertical axis and residues were displayed on the horizontal axis. Residues making accumulated contacts shorter than 2 ns are abandoned.

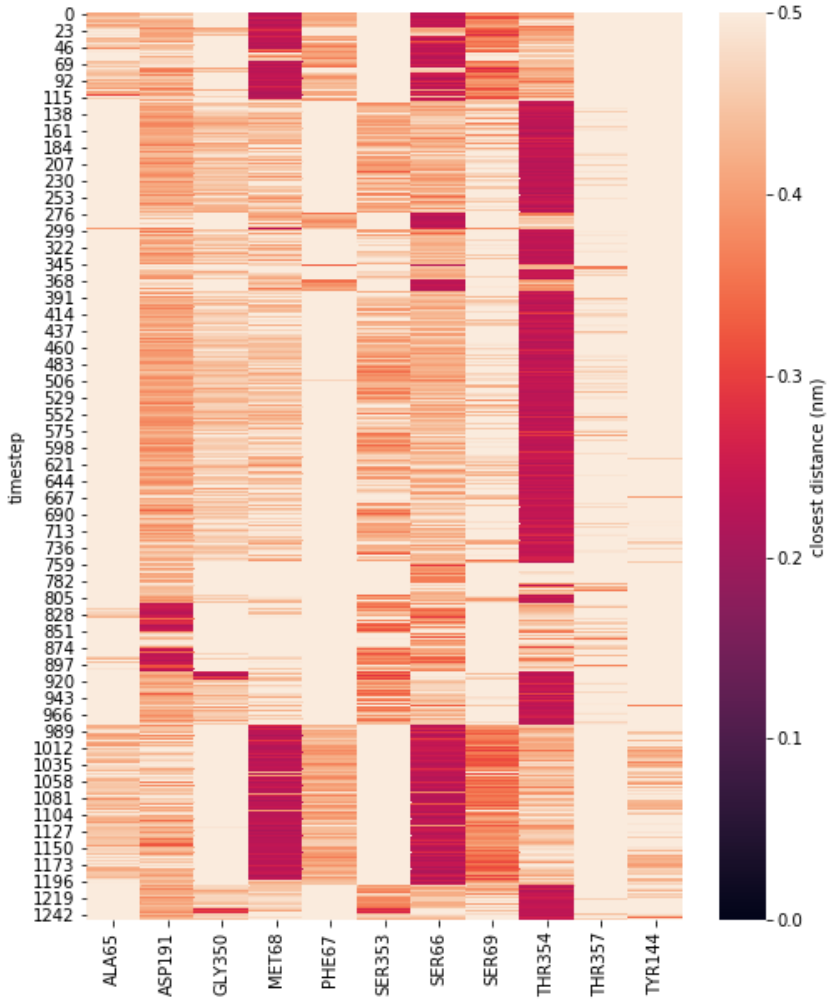

Figure 3.8: Frequent contacts between NaB and residues. Frame indices are plotted on the vertical axis and residues were displayed on the horizontal axis. Residues making accumulated contacts shorter than 2 ns are abandoned.

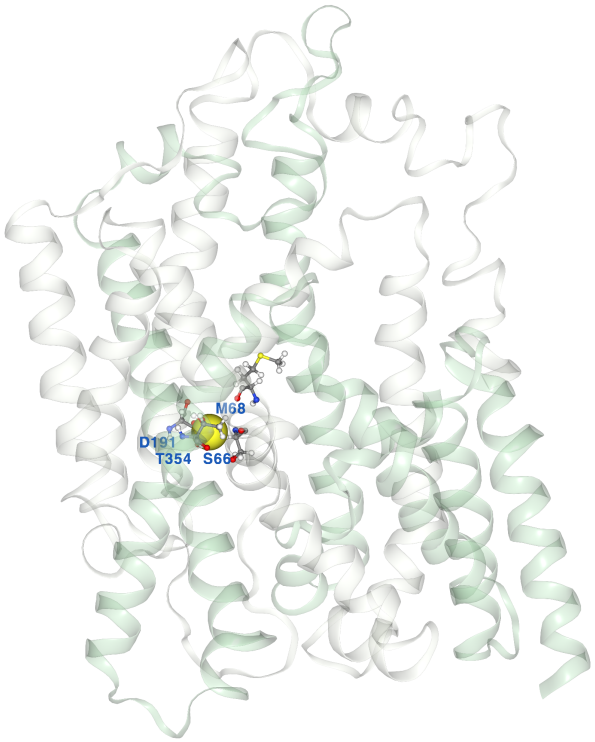

Figure 3.9: Na2 site. NIS is represented in Cartoon and key residues composing Na2 Site are represented as balls and sticks.

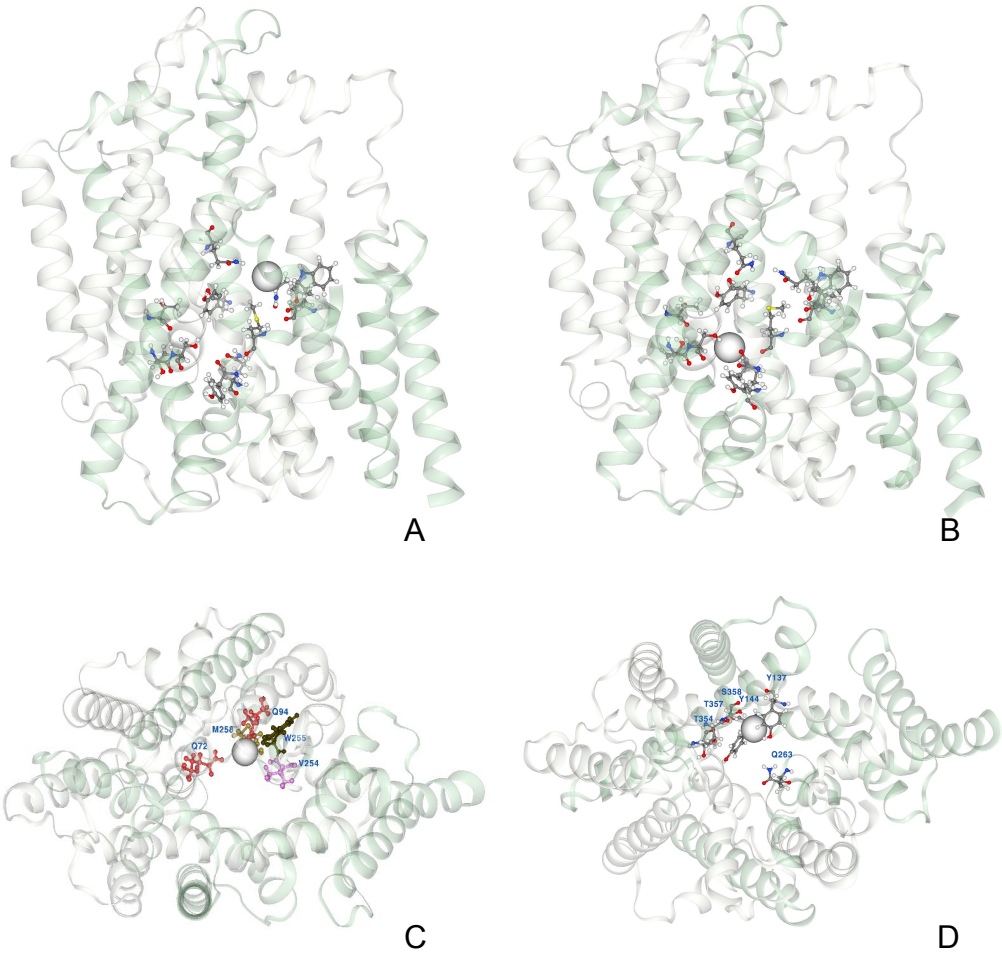

Figure 3.10: I^- binding pockets. NIS is represented in Cartoon and key residues composing the I^- site are represented as balls and sticks. The I^- is represented as the white hyperball. Figure A and B provides side views of the protein as well as those key residues. Figure C and D represent the perspective from the extracellular side to the intracellular side

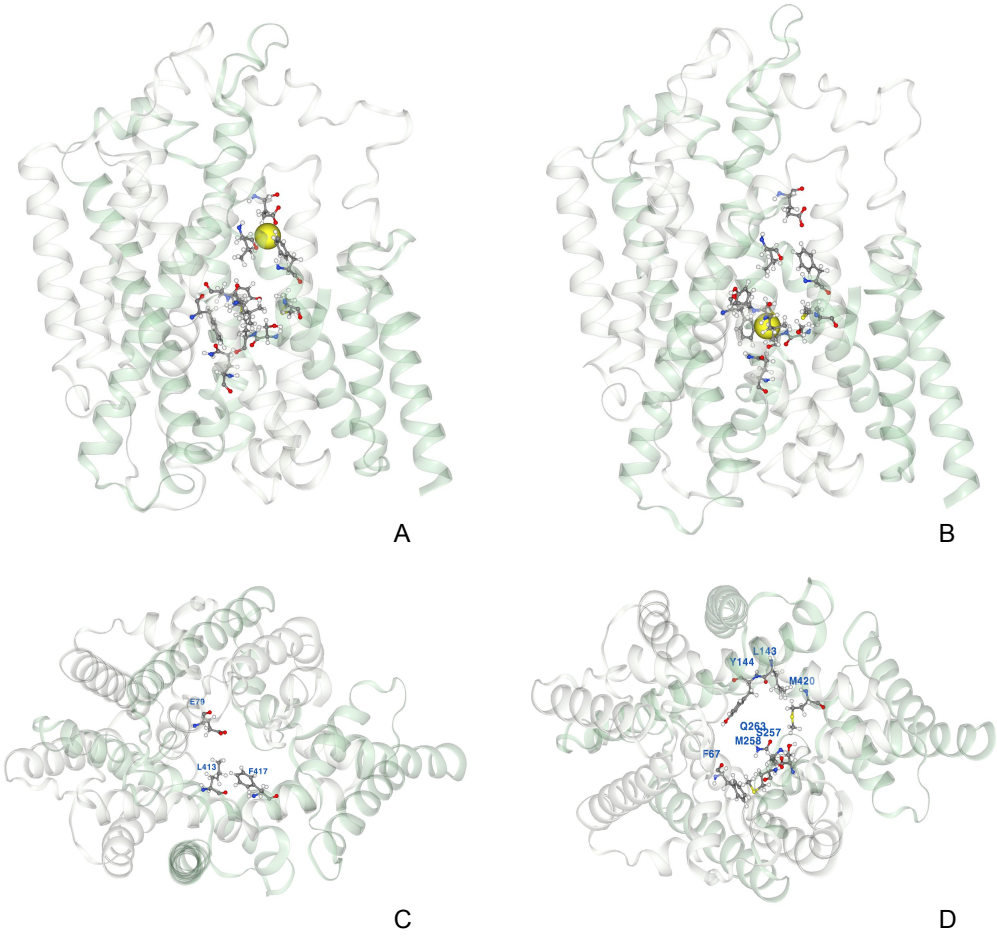

Figure 3.11: NaA binding pockets. NIS is represented in Cartoon and key residues composing the binding pockets are represented as balls and sticks. The Na1 is represented as the yellow hyperball. Figure A and B provides side views of the protein as well as those key residues. Figure C and D represent the perspective from the extracellular side to the intracellular side

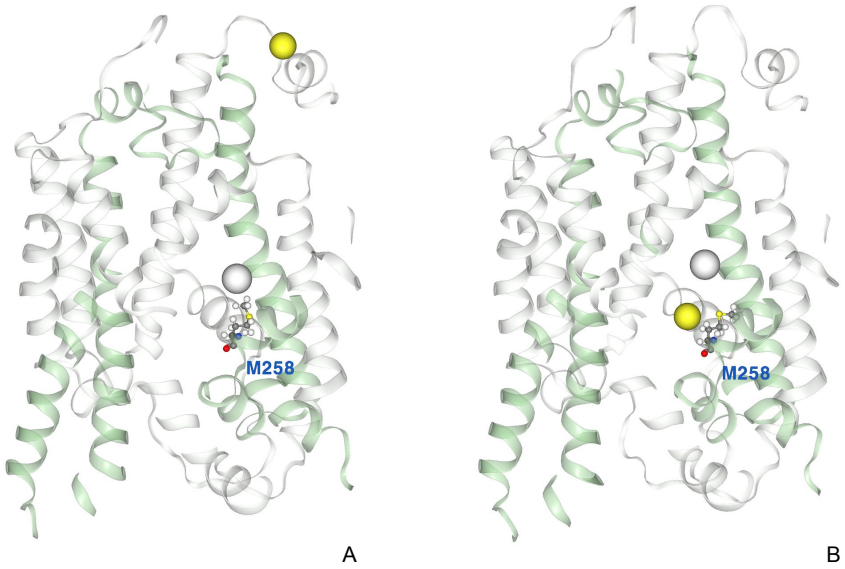

Figure 3.12: Side chain orientations of M258. Figure A plots the extracellular side pointing orientation of M258 side chain when I^- binds the upper pocket. Figure B shows the M258 side chain becomes flat after NaA moves throught the upper pocket. The protein is shown in Cartoon. Na^+, I^- are shown in yellow and white hyperballs, respectively. M258 are represented with balls and sticks. The protein is clipped near to avoid blocking M258.

References

[1] Carla Portulano, Monika Paroder-Belenitsky, and Nancy Carrasco. "The Na+/I-symporter (NIS): mechanism and medical impact". In: *Endocrine reviews* 35.1 (2013), pp. 106–149.

[2] Nancy Carrasco. "Iodide transport in the thyroid gland". In: *Biochimica et Biophysica Acta (BBA)-Reviews on Biomembranes* 1154.1 (1993), pp. 65–82.

[3] Christoph Reiners, Heribert Hänscheid, Markus Luster, Michael Lassmann, and Frederik A Verburg. "Radioiodine for remnant ablation and therapy of metastatic disease". In: *Nature Reviews Endocrinology* 7.10 (2011), p. 589.

[4] Garcilaso Riesco-Eizaguirre and Pilar Santisteban. "A perspective view of sodium iodide symporter research and its clinical implications". In: *European Journal of Endocrinology* 155.4 (2006), pp. 495–512.

[5] M Hingorani, C Spitzweg, G Vassaux, K Newbold, A Melcher, H Pandha, R Vile, and K Harrington. "The biology of the sodium iodide symporter and its potential for targeted gene delivery". In: *Current cancer drug targets* 10.2 (2010), pp. 242–267.

[6] Alan R Penheiter, Stephen J Russell, and Stephanie K Carlson. "The sodium iodide symporter (NIS) as an imaging reporter for gene, viral, and cell-based therapies". In: *Current gene therapy* 12.1 (2012), pp. 33–47.

[7] Ge Dai, Orlie Levy, and Nancy Carrasco. "Cloning and characterization of the thyroid iodide transporter". In: *Nature* 379.6564 (1996), p. 458.

[8] Pavlos Msaouel, Ianko D Iankov, Cory Allen, Ileana Aderca, Mark J Federspiel, Donald J Tindall, John C Morris, Michael Koutsilieris, Stephen J Russell, and Evanthia Galanis. "Noninvasive imaging and radiovirotherapy of prostate cancer using an oncolytic measles virus expressing the sodium iodide symporter". In: *Molecular Therapy* 17.12 (2009), pp. 2041–2048.

[9] Juan P Nicola, Nancy Carrasco, and L Mario Amzel. "Physiological sodium concentrations enhance the iodide affinity of the Na+/I- symporter". In: *Nature communications* 5 (2014), p. 3948.

[10] Giuseppe Ferrandino, Juan Pablo Nicola, Yuly E Sánchez, Ignacia Echeverria, Yunlong Liu, L Mario Amzel, and Nancy Carrasco. "Na+ coordination at the Na2 site of the Na+/I- symporter". In: *Proceedings of the National Academy of Sciences* 113.37 (2016), E5379–E5388.

[11] Antonio De la Vieja, Orsolya Dohan, Orlie Levy, and Nancy Carrasco. "Molecular analysis of the sodium/iodide symporter: impact on thyroid and extrathyroid pathophysiology". In: *Physiological reviews* 80.3 (2000), pp. 1083–1105.

[12] Sepehr Eskandari, Donald DF Loo, Ge Dai, Orlie Levy, Ernest M Wright, and Nancy Carrasco. "Thyroid Na+/I- symporter Mechanism, stoichiometry, and specificity". In: *Journal of Biological Chemistry* 272.43 (1997), pp. 27230–27238.

[13] Orsolya Dohan, Antonio De la Vieja, Viktoriya Paroder, Claudia Riedel, Mona Artani, Mia Reed, Christopher S Ginter, and Nancy Carrasco. "The sodium/iodide symporter (NIS): characterization, regulation, and medical significance". In: *Endocrine reviews* 24.1 (2003), pp. 48–77.

[14] Mark James Abraham, Teemu Murtola, Roland Schulz, Szilárd Páll, Jeremy C Smith, Berk Hess, and Erik Lindahl. "GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers". In: *SoftwareX* 1 (2015), pp. 19–25.

[15] Jing Huang and Alexander D MacKerell. "CHARMM36 all-atom additive protein force field: Validation based on comparison to NMR data". In: *Journal of computational chemistry* 34.25 (2013), pp. 2135–2145.

[16] Sunhwan Jo, Taehoon Kim, Vidyashankara G Iyer, and Wonpil Im. "CHARMM-GUI: a web-based graphical user interface for CHARMM". In: *Journal of computational chemistry* 29.11 (2008), pp. 1859–1865.

[17] Schrödinger, LLC. "The PyMOL Molecular Graphics System, Version 1.8". 2015.

[18] Alexander S Rose and Peter W Hildebrand. "NGL Viewer: a web application for molecular visualization". In: *Nucleic acids research* 43.W1 (2015), W576–W579.

[19] Weixiao Y Wahlgren, Elin Dunevall, Rachel A North, Aviv Paz, Mariafrancesca Scalise, Paola Bisignano, Johan Bengtsson-Palme, Parveen Goyal, Elin Claesson, Rhawnie Caing-Carlsson, et al. "Substrate-bound outward-open structure of a Na+-coupled sialic acid symporter reveals a new Na+ site". In: *Nature communications* 9.1 (2018), p. 1753.

Chapter 4

Large-Scale Structure-Based Virtual Screening

In general, Virtual Screening (VS) refers to an *in silico* technique for identifying small molecule compounds which potentially bind non-covalently to a protein target with high affinites. Compared with its experimental counterpart, namely high-throughput screening (HTS), it is proven that [1] VS is faster and more cost-effective for scanning large compound databases, while yielding a poorer yet empirically acceptable result. Due to these advantages, VS has gained interest from both pharmaceutical companies and research labs for searching drug-like compounds or inhibitors for experimental assays.

For protein targets for which atomic structures or homology models are available, VS can be accomplished by the following workflow: docking small compounds into the binding sites of the structures or a defined search space and ranking all the ligands by their predicted binding affinities. The docking step in the workflow refers to predicting the binding affinity and bound conformations of every compound in question by optimizing a pre-defined, empirically parameterized scoring function in the conformation space of the system. VS techniques involving such a workflow are usually categorized as Structure-Based Virtual Screening (SBVS). From a Structured-Based Drug Design (SBDD) perspective, the predicted bound conformations provide direct information about the interactions of compounds with the protein at an atomic level and can serve as skeletons for further drug refinement. In our studies, we used the available structures of our protein targets and adopted SBVS as the technical basis for screening large-scale compound databases.

4.1 AutoDock Vina

Molecular docking, the core step of SBVS, has been modeled and implemented by various research groups [2]. Among all available softwares, Autodock Vina (Vina) [3] was cited over 6000 times since it was pubished in 2010. It defines a general form of a scoring function for any system consisting of a single compound and a protein target as

$$s = s_{inter} + s_{intra} = \sum_{i<j} f_{ij}(r_{ij}) \tag{4.1}$$

where s denote the score, which could be contributed from both intermolecular interactions s_{inter} and intramolecular interactions s_{intra}, f_{ij} represents a non-linear function specifically depending on the atom pair i and j in the system that maps the distance between the pair to a "score". This set of functions are determined by both knowledge-based potentials and empirical parameter optimizations. Intuitively, f should be able to describe interactions such as hydrophobic interactions, electrostatic interactions, van der waals interactions, etc.

The scoring function is essentially an attempt to describe the chemical potentials of the compound and its interaction with the protein, whereas the free energy of binding only involves the intermolecular interactions. Docking cares about the free energy of binding which usually serves as a metric of the binding affinity. To bridge between the defined scoring function and the free energy of binding, Vina uses an abitrary strictly increasing smooth nonlinear function g to map the s_{inter} to the predicted free energy of binding, since free energy of binding only concerns about the intermolecular interactions.

With the scoring function settled, Vina finds the score-minimized conformations by running the following algorithm:

1. Initially, it samples a few random conformations. The number of the sampled conformations is determined by the number of poses requested by users.

2. For each sampled conformation, it optimizes the starting point by a gradient based optimizer that alternates between a change in conformation and a local optimization step.

3. It keeps a set of significant score-minimized structures and returns them with their the estimated free energy of binding.

The implementation of Vina in C++ is very efficient since it involves both algorithm-level optimizations and concurrency programming. It reads both the compound and the protein target in a PDBQT file format, which is used for AutoDock 4 [4] and can be considered as an extension of the PDB file format. In addition, the user interface of Vina is quite friendly. Unlike AutoDock 4, it offers a single commmand line interface that takes a compound, a protein target and a defined search space, and outputs n lowest free energy of binding together with the corresponding conformations. According to an evaluation of multiple docking programs [2], Vina ranks high among all open-source distributed academic softwares. It is also worth mentioning that researchers have been making efforts to further improve the computational performance of Vina for years. Many optimizations of Vina [5] [6] were published recently but most of them lack evidence of their effects based on a large number of empirical cases. Despite this fact, we consider Vina to be an appropriate tool to be utilized for our studies.

4.2 SparkVina

A simple approach for running SBVS with Vina is documented in its manual [7]. The approach is just a simple bash script that loops over all ligand files and process them one by one. This approach is too naive to support a million-scale compound database for the following reasons:

1. The database needs to be diced into millions of files. Reading and processing all those files significantly increase the I/O overheads.

2. Though Vina could utilize multiple cores with its multithreading support, it can hardly be deployed in a distributed cluster. In addition, we have to mention that when fixing the number of modes, requesting more cores than the number of modes will not further decrease the running time because Vina maps the optimization of one mode to at most one core.

3. For large-scale virtual screening, most of the ligands do not dock well so it is not necessary to write their docked conformations.

To support scanning large compound databases, it is necessary to improve the implementation of Vina to make it scalable. A natural idea to accomplish this is to wrap the current Vina code with some existing parallel computing and distributed data processing frameworks, such as MPI [8] or Apache Spark [9]. We didn't consider CUDA since translating Vina logic to GPUs needs almost a rewrite of Vina, which was left as a future direction. Since Vina is written in C++, the MPI C++ framework seems to be well fit for this goal. However, MPI programs cannot scale up and down elastically and other data transformation steps have to be manually implemented. In contrast to MPI, Apache Spark provides excellent high-level abstractions on parallel data transformations and its master-slave architecture enables our software to run on any complicated distributed environment, for instance, a distributed cluster or a commercial cloud computing facility. Thus, we settled with Apache Spark and we named our software **SparkVina**.

4.2.1 Python Interface for Vina

Unfortunately, Apache Spark was written in Scala and launched on Java Virtual Machine (JVM). Spark choosed Scala as the programming language for its implementation since it originated as a research project at UC Berkeley so the authors could aim at fast development with a concise and cross-platform language. Even now the full functionality of Spark could only be accessed through its Java or Scala API. To be compatible with more projects, developers of Apache Spark provides a Python API, namely PySpark, by communicating with Java bytecode through the Py4J library.

Since the code base of Vina is in C++, we needed to adapt it either to JVM through the Java Native Interface (JNI) library or to Python. Compared with a Java interface for Vina, a Python interface can offer more convenience in interactive tasks, for instance, running molecular docking and analyzing or visualizing the result all in one Jupyter notebook. In practice, bridging to Python via Swig tools is easier to operate and to open it for extensions, i.e., the Swig interface file doesn't need any modification when

extending the interface. Therefore, rather than building a Java interface for Vina, we modified several pieces of high level C++ code of Vina added a light-weighted Python interface for it with Swig 3.0.

To reduce the I/O burden of our program and the fragmentation of the file system, we designed an utility function that can read from a file that contains a collection of compounds, known as a shard. In practice, many large compound databases, such as, Zinc [10], distribute data in shards which are actually gunzip compressed files of collections of compounds. We designed our implementation to support reading both gunzipped shards and uncompressed shards. The function dices the shard to a list of compound PDBQT strings and the compound PDBQT strings are transmitted across the multi-nodes network.

A serious problem with Vina's implementation is that when a protein-ligand system is constructed, the bound ligand cannot be removed from the system object so when docking a new compound, a copy of the original parsed protein model is made. However, we consider that the overhead of making copies is negligible, compared with the cost of docking.

We implemented a `VinaDock` class similar to the original Vina's command line interface. An example of using the Python API we added is shown below.

```
import fiesta3.spark_vina.vina_wrap as vina

num_cpus = 4
exhaustiveness = 5

# filters out all candidates whose predicted
# free energy of binding is greater than a threshold.
threshold = 1.0

model = vina.VinaDock(
  'protein.pdbqt',
  0, 0, 0,     # 3-d coordinate of the center of the search box
  30, 30, 30,  # the size of the search box.
```

```
  num_cpus,
  exhaustiveness)

results = model.vina_fit(
  vina.read_ligand_to_strings('ligands.pdbqt.gz'),
  threshold)
```

4.2.2 Data Processing Workflow

After integrating Vina with Python, it is trivial to implement the entire SBVS logic with a simple Resilient Distributed Datasets (RDD) [11] transformation flow, as shown in Figure 4.1. All compound shards are parsed in parallel to a parallel list of PDBQT strings. We added a repartition step to load-balance the number of compounds assigned to each Spark executor. Then each executor runs docking for the list of compounds assigned. We finally sort the results and take the first n compounds with the lowest predicted free energies of binding.

4.2.3 Running Configurations

In our implementation of SparkVina, we don't make any default configurations for the runtime behaviors. This is because we desire to completely separate our Python deliverables with its runtime configurations. All runtime configurations should be passed to `spark-submit` directly. We enforce this rule just because sometimes configurations set inside the program have the highest priority [12] to be acknowledged by the Spark context manager. And apparently changing source code for merely changing the hard coded configurations is a bad programming practice. Here is an example of passing configurations to `spark-submit`.

```
spark-submit                                    \
  ...                                           \
  --conf "spark.executor.instances=14"          \
  --conf "spark.local.dir=/local/"              \
  --conf "spark.task.cpus=4"                    \
  --conf "spark.executor.cores=4"               \
```

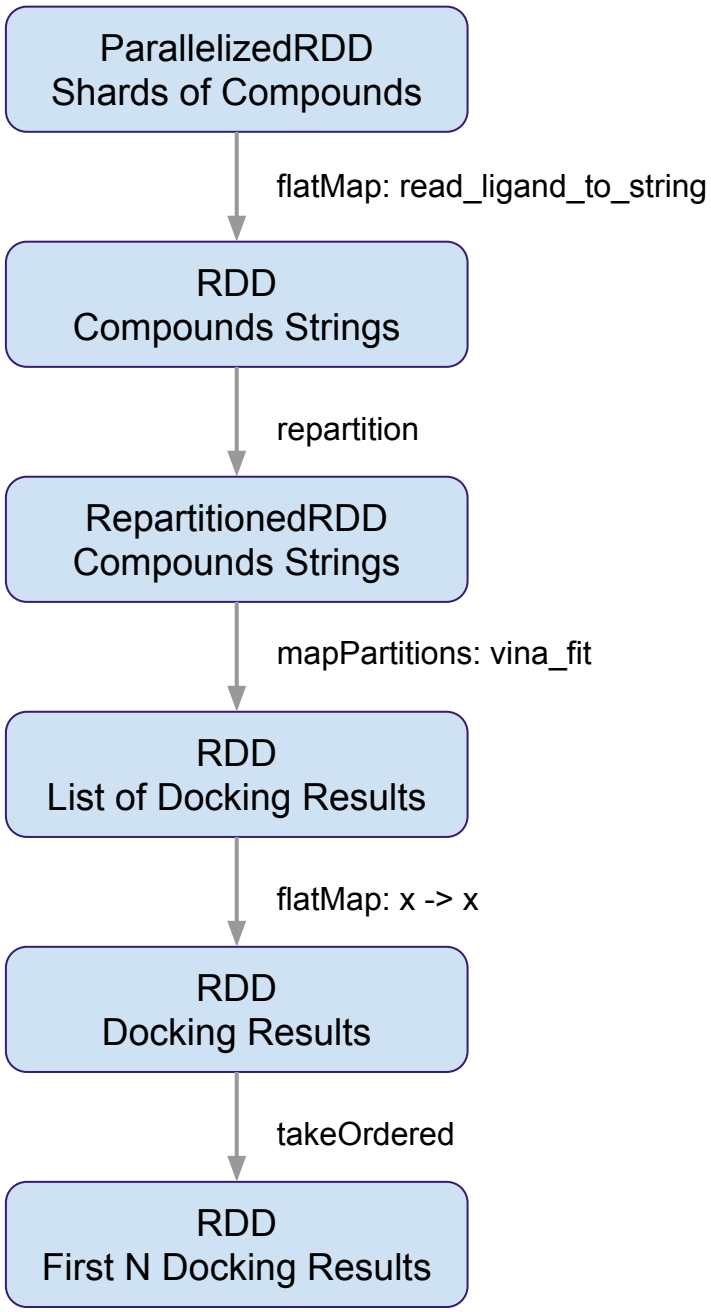

Figure 4.1: RDD (Resilient Distributed Datasets) transformation flow implementing SBVS

```
  --conf "spark.driver.cores=4"                    \
  --conf "spark.executor.memory=8g"                \
  ...
```

4.3 Experiments

4.3.1 SBVS Experiment

In response to a collaboration requested by Dr. Gregg Semenza's research group (Johns Hopkins University School of Medicine, Department of Pediatrics), we launched a large-scale SBVS experiment that searches effective inhibitors for the transcription factor, Hypoxia-inducible factor 2-alpha (HIF-2α).

HIF-2α is the second isoform of the α-subunit of HIF. It is a heteordimer which consists of two transactivation domains. Crystal structures of HIF-2α bound with proflavin (PDB code: 4ZPH) [13] clearly shows that the small compound binds to the interface between the heterodimer, which disrupts the HIF dimerization and inhibits its transcription activity. Unfortunately, proflavin is an antiseptic dye that can intercalate ds-DNA and not a drug-like compound. Thus, the goal of this experiment is to identify drug-like inhibitors that bind to the interface of HIF-2α through screening a large compound database with our SBVS technique.

We use the crystal structure 4ZPH as our protein target in our experiment. The ligand proflavin was removed manually from 4ZPH to make the interfacial pocket void and then the rest of the PDB file was converted to PDBQT format with utility tools provided by AutoDock 4 and AutoDockTools 4 [4]. We manually defined the search box centered in the original binding site of proflavin with a size of 30 $\times$ 30 $\times$ 30. We screened all compounds in the cells listed in Figure 4.2 of the ZINC 15 database [14] which includes 220K compounds and we collected the top 100 compounds with the lowest predicted free energy of binding. Figure 4.3 shows the bound conformation of an identified compound (ZINC000009267078) with a predicted binding free energy of -9.6 kcal/mol, which is verified by multiple trials with different random seeds. In the figure, we can see the compound (in orange) bound to the same site that proflavin (in yellow)

	Molecular Weight				
LogP		375	400	425	450
	0	FB	GB	HB	IB
	1		GC	HC	IC
	2		GD		ID
	2.5				IE
All the cells are marked with "3D" and "in stock" label. The total number of compounds is ~220K.					

Figure 4.2: Compounds docked to the HIF-2α.

binds in the original crystal structure.

The experiment was conducted with two computational nodes with 56 Intel Xeon CPUs of MARCC (See Chapter 1), which was the maximum number of CPUs that were allowed to use. It takes approximately 5 days to run and the estimated throughput is 45K compounds per day. The throughput can be easily increased as more computational resource become available. (See the next experiment) A detail configuration of this job is available through `//spark_vina/hif2a.slurm` (See Chapter 1 for the structure of our code repository).

4.3.2 Scalability

To better understand the scalability of SparkVina, we did a small experiment with the system described in the previous section. We use a small compound dataset, which is actually one medium-size shard including 68 compounds. We ran four separate experiments with SparkVina to screen this specific shard using 4, 8, 16, and 32 CPUs,

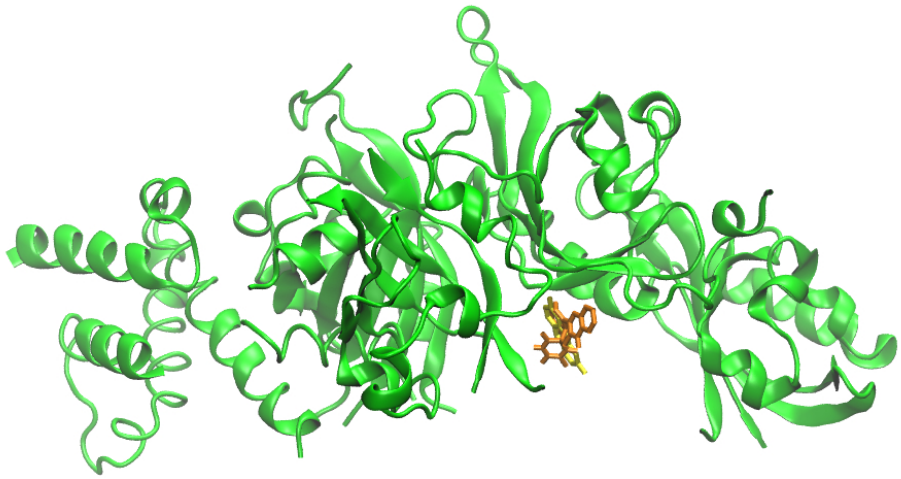

Figure 4.3: The binding conformation of the "best" compound we identified

respectively. Figure 4.4 shows the running time for each experiment and Figure 4.5 plots the scalability of SparkVina.

It turns out that SparkVina scales well. In the case of requesting m modes, where m is a constant, the original vina 's implementation can only make use of at most m cores while in principle, our SparkVina can use as many as possible.

4.4 Future Work

Future work on improving the performance of SparkVina may include optimizations of two aspects:

1. Improving the accuracy and efficiency of the current Vina algorithm.

2. Shortening the average running time for docking one compound.

The first aspect requires a large collection of accurate binding data coming from experiments since the parameters of Vina's scoring function (Equation 4.1) was trained in

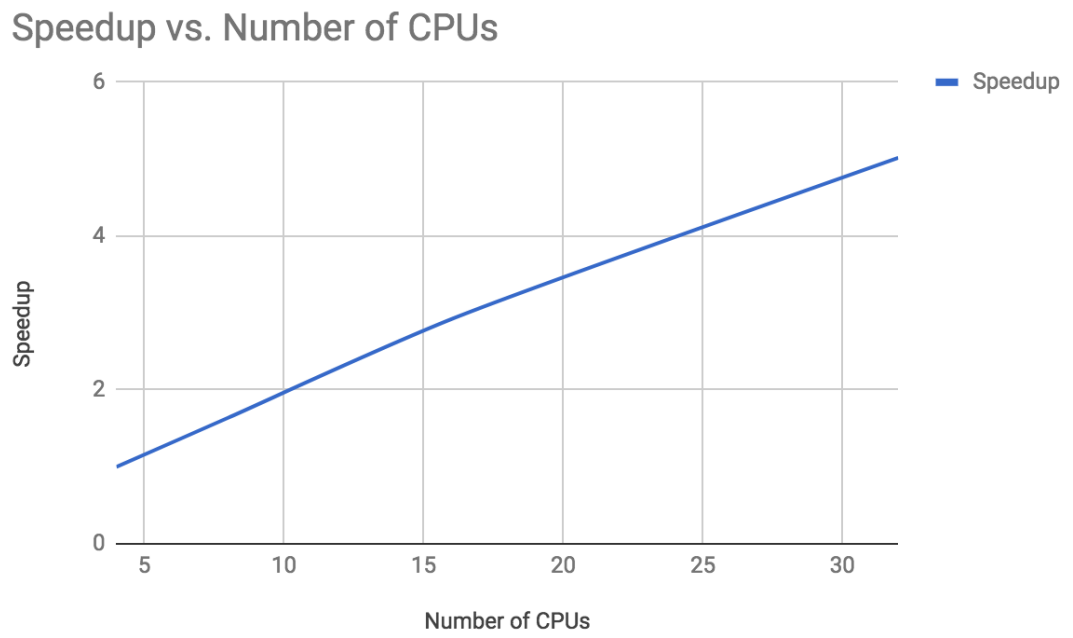

Figure 4.4: Running Time vs. Number of CPUs

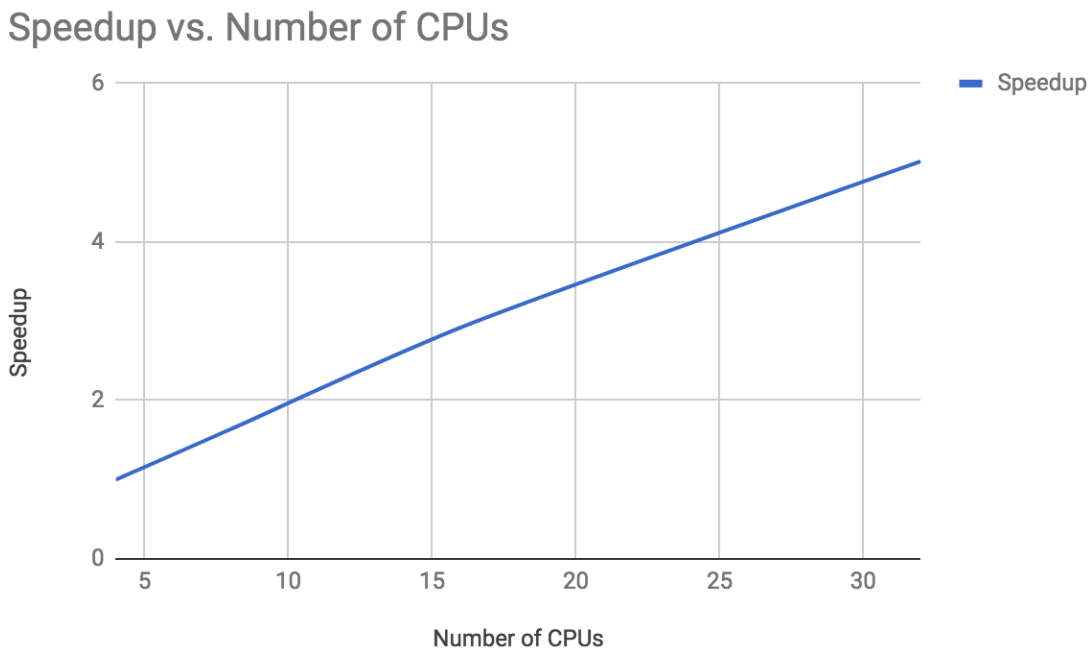

Figure 4.5: Speedup vs. Number of CPUs

a "non-linear regression" style. We can alter the regression model of the scoring function by including higher power terms of r_{ij} with regularization since the approximation of the chemical potentials of a complicated multi-body systems does usually fall into the expectation of classical physics. To avoid exaggerating the size of the parameter space by interpolation, we can do a pruning step that eliminates all terms with very little effects. Other options on constructing a more effective model could be attempted as well. However, acquiring a collection of accurate training data remains a big challenge.

The second aspect could be accomplished in two ways. Improving the running time of Vina at an algorithm level usually provides a great enhancement. A direct idea would be replacing the current optimizer of Vina with one converging faster. This approach usually needs far more investigations than improving running time at the implementation level. Optimizing the old implementation of Vina to achieve performance enhancement is a task that could be considered in many directions, such as restructuring the old Vina implementation, removing redundant copies, involving cache-friendly programming and adding Single Instruction Multiple Data (SIMD) acceleration. In general, improving the running time of Vina could significantly enlarge the number of compounds scanned so that more potential ligands can be found and tested with experiments.

References

[1] E. Lionta, G. Spyrou, D. K. Vassilatis, and Z. Cournia. "Structure-Based Virtual Screening for Drug Discovery: Principles, Applications and Recent Advances". In: *Curr Top Med Chem.* 14.16 (2014), pp. 1923–1938.

[2] N. S. Pagadala, K. Syed, and J. Tuszynski. "Software for molecular docking: a review". In: *Biophys Rev.* 9.2 (2017), pp. 91–102.

[3] O. Trott and A. J. Olson. "AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading". In: *J Comput Chem.* 31.2 (2010), pp. 455–461.

[4] G. Morris, R. Huey, W. Lindstrom, M. F. Sanner, R. K. Belew, D. S. Goodsell, and A. O. Olson. "Autodock4 and AutoDockTools4: automated docking with selective receptor flexiblity". In: *J Comput Chem.* 30.16 (2009), pp. 2785–2791.

[5] A. Alhossary, S. D. Handoko, Y. Mu, and C. K. Kwoh. "Fast, accurate, and reliable molecular docking with QuickVina 2." In: *Bioinformatics.* 31.13 (2015), pp. 2214–6.

[6] R. Quiroga and M. A. Vilarreal. "Vinardo: A Scoring Function Based on Autodock Vina Improves Scoring, Docking, and Virtual Screening". In: *PLoS One.* 11.5 (2016).

[7] *AutoDock Vina Manual (Section Virtual Screening).* URL: `http://vina.scripps.edu/manual.html#screening`.

[8] Message Passing Interface Forum. "MPI: A Message-Passing Interface Standard". In: *Message Passing Interface Forum* (1994).

[9] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. "Spark: Cluster Computing with Working Sets". In: *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing.* HotCloud'10. Boston, MA: USENIX Association, 2010, pp. 10–10. URL: `http://dl.acm.org/citation.cfm?id=1863103.1863113`.

[10] J. J. Irwin, T. Sterling, M. M. Mysinger, E. S. Bolstad, and R. G. Coleman. "ZINC: A Free Tool to Discover Chemistry for Biology". In: *J. Chem. Inf. Model.* 52.7 (2012), pp. 1757–1768.

[11] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. "Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing". In: *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation.* NSDI'12. San Jose, CA: USENIX Association, 2012, pp. 2–2. URL: `http://dl.acm.org/citation.cfm?id=2228298.2228301`.

[12] *Spark Configuration.* URL: `https://spark.apache.org/docs/latest/configuration.html`.

[13] D. Wu, N. Potluri, J. Lu, Y. Kim, and F. Rastinejad. “Structural integration in hypoxia-inducible factors”. In: *Nature* 524 (2015), pp. 303–308.

[14] T. Sterling and J. J. Irwin. “ZINC 15 – Ligand Discovery for Everyone”. In: *J. Chem. Inf. Model.* 55.11 (2015), pp. 2324–2337.

Chapter 5

Discussion and Conclusions

MD simulations have been widely recognized as pivotal tools for sampling molecular dynamics of biological macromolecules in the community of biophysics since the last century. Technical improvements of MD tremendously increase the lengths of trajectories, which allows us exploring the larger scope of the conformational space of macromolecules and makes understanding the link between the conformational states and protein functions possible. In the case of NIS we presented, MD trajectories with an accumulated length of $\sim 4\mu$s provided us valuable insights into how two Na^+ and an I^- are co-transported by NIS. In addition to the conventional MD simulations running in equilibrium, we adopted position restraint simulations to explore conformational changes quickly, which turns out to be fairly effective. The restraint induced conformation allows ions binding, which grants ion transport partially being observed in one of the extended simulations after position restrains were removed. Unfortunately, because of the limited computational resources, unbiased sufficient sampling of NIS was not attempted and thus estimating transitional probabilities between any two pairs of conformational states was not possible. Running multiple groups of conventional simulations, each of which is comprised of simulation replicas starting from a specific conformation, could be considered as a major task in the next phase of this work.

Being only capable of acquiring long dynamics of proteins is not enough for understanding protein mechanisms. Effective analytical methods should be applied to the data that we collect to fetch the underlying information of our interest. As we introduced in previous chapters, various type of unsupervised machine learning algorithms

are adopted by computational biophysicists to analyze the patterns of motion and to cluster frames based on their geometrical coordinates. We contributed a novel, data-driven and generative clustering method, AAE-GS, based on a popular deep learning method, AAE. We showed the power of AAE-GS by using it to cluster long MD trajectories of two different proteins, Trp-Cage and NIS. In the case of Trp-Cage, AAE-GS accurately clusters more than a million frames into 8 clusters, each of which has its unique set of characteristics. Since the Trp-Cage trajectory was sufficiently sampled, we assumed the Markov assumption and were able to build a highly interpretable folding pathway of Trp-Cage based on the clustering results. In the other case of NIS, AAE-GS were able to identify the source of a frame, which proves that AAE-GS can detect subtle distinct features lying in each trajectory. Those subtle characteristics cannot be usually distinguished by human observation. From our actual experiences on applying machine learning methods to MD data, we found they are extremely valuable and that they have the potential to solve problems raised by other biophysical systems.

Last but not the least, we showed the power of big data and computational structural biology in screening drugs. We modified the original implementation of Vina to make it compatible with the interface of PySpark, which finally yields a scalable structural virtual screen software, SparkVina. SparkVina is capable of being scaled to multiple high-performance computing nodes and screening millions of compounds per day. We successfully used SparkVina to screen drugs that hinder the dimerization of HIF-2α. According to our collaborator, Dr. Gregg Semenza, multiple potential candidates that we provided work in their experiments. Improvements from multiple aspects, such as using reinforcement learning to enhance the proportion of experimentally working candidates rate and optimizing the current software implementations, can be pursued in the next phases of this work.

In conclusion, the research work presented in this dissertation shows that computational approaches are critical in understanding the working mechanisms of the protein as well as resolving important biophysics-related, application-level tasks.

Appendix A

Understanding the mechanism of physiological effectors and oncogenic mutations in the activation of PI3K: A computational approach

A.1 Introduction

Class IA PI3Ks are classified into three different isoforms, PI3Kα, PI3Kβ and PI3Kδ. They are heterodimers composed of a regulatory (p85) and a catalytic domain (p110). PI3Ks are among only a small number of kinases that function as lipid kinases: they take the γ-phosphate of ATP, transfer it to the 3 position of PIP2 and generate PIP3 [1]. This chemistry is carried out by the catalytic subunit of PI3K, p110, and the activity reaches maximum after the nSH2 domain of the regulatory subunit binds to its physiological effector, peptides within the C-terminus of Receptor Tyrosine Kinases (RTKs) that contain a phosphorylated tyrosine. The product of this enzymatic reaction, PIP3, serves as a docking site for many downstream proteins that contain a pleckstrin homology domain (PH domain) and act as key regulators of many important cellular pathways [1], such as AKT and PDK1 that broadly regulate cell proliferation, survival and cell death [2–5].

In addition to their pivotal function in signal transduction, PI3Ks play an important role in diseases, especially in cancers. The first major evidence that showed the involvement of PI3K in cancers came from the analysis of the genome of Avian Sarcoma Virus 16

(ASV16). This analysis showed that ASV16 genome contains an oncogene derived from a cellular gene, PI3KCA, encoding the catalytic subunit of PI3Ks [6]. Most recently, it has been reported that PI3KCA, was somatically mutated in diverse types of cancers, including colon, rectum, breast, liver, brain and ovary [2–5, 7–12]. The COSMIC database shows that most of these mutations (∼99%) found in cancer patients occurred in the p110 subunit of PI3Ks. These cancer mutations render the kinase constitutively active making PI3Ks ideal targets for novel cancer drugs [13].

Atomic resolution structures of PI3Ks are key elements in structure-based and mutation-specific novel drug design. In the atomic X-ray structure of PI3Kα (p110α in complex with the nSH2 and the iSH2 domains of p85α) published in 2007 [14], all five different domains of the p110α subunit can be clearly identified: an N-terminal ABD domain (Adaptor Binding Domain) which binds to the regulatory p85 family members, a Ras-binding domain (RBD), a C2 domain, a helical domain and the kinase domain. Of the p110α binding partner p85α, only nSH2 (N-terminal SH2 domain) and iSH2 (inter-SH2 domain) are present in the structure (Figure A.1). This construct shows the locations of all oncogenic mutations.

Through the X-ray crystal structures of Class I PI3Ks, including atomic structures of other isoforms (PI3Kγ and PI3Kδ), have been determined and well studied in recent publications, the molecular mechanism of PI3Ks activation by its physiological effectors and oncogenic mutations still remains unknown. Previous experimental results merely showed that in the basal state of PI3K, p85 regulatory subunits bind to and inhibit the p110α catalytic subunit; however, when responding to specific cellular stimuli, the nSH2 and cSH2 (not shown in the crystal structure) domains bind phosphorylated tyrosine residues of the C-terminus activated receptors or adaptor proteins, and this phosphotyrosine binding activates the p110α catalytic subunits. It is worthwhile to note that after PI3K gets activated, the p85α subunit doesnt dissociate from p110α and the heterodimer state persists, and is required for catalytic activity.

To understand the activation mechanism of PI3K, we used computational approaches, such as MD simulations, to tackle this problem. Our previous computational research

[15] conducted multiple implicit-solvent simulations to show that multiple oncogenic mutations destabilize the p110α-p85α, which are far from the catalytic core of the protein and have the effect of increasing the enzymatic affinity. By affecting the dynamics of the protein, these mutations favor the conformations that reduce the autoinhibitory effect of the p85α nSH2 domain. Analysis of the interdomain interactions of the wild-type and mutants at the p110αp85α interface obtained with molecular dynamics simulations suggest that all of the tumor-associated mutations effectively weaken the interactions between p110α and p85α by disrupting key stabilizing interactions. In this work, we carried out long explicit-solvent simulations to further investigate the activation mechanism of PI3K under physiological conditions and the effects of specific oncogenic mutations which locate remote from the catalytic site on the activation of the enzyme.

A.2 Methods

A.2.1 The Construction of MD Simulation System

The crystallographic structures of WT p110α in complex with the iSH2 domain of p85 a [14] (PDB ID: 2RD0) and the H1047R mutant of p110α in complex with the niSH2 domain of p85a [16] (PDB ID: 3HIZ) were used to build the different models. Related structures of the free WT and the WT in complex with the lipid substrate have been recently published (PDB ID: 4OVV) [17]. All crystal structures lack some loop regions, which were built with the loop-building option of MODELLER 9v8 [18]. One loop of critical importance that needed to be modeled was the activation loop (p110α residues 933957). This loop was modeled in the inactive conformation with the activation loop of the PI3Kβ isoform as a template [19]. We modeled the WT and two mutants of p110α, the ABD mutations R38CR88Q and a combination of ABD mutations and C2 mutation R38CR88QN345K, in complex with the nSH2 and iSH2 domains of p85α. The mutants were modeled using PYMOL. To involve ligands of the PI3K in the simulation systems, we aligned our full-length model with 1) a crystal structure with lipid substrates (PDB ID: 4OVV) to place two $diC4-PIP_2$ (PBU) ligands and 2) a crystal structure of PI3Kγ (PDB ID: 1E8X) to place the ATP and 2 Mg^{2+} ions.

We adopted the CHARMM22* force field [20] for running all simulations. Since ATP and PBU are not regular molecules whose force field parameters are included in CHARMM22*, we used a force field generator tool SwissParam[21] to generate those parameters for the CHARMM force field for ATP and PBU. Then we use GROMACS 2016.4 to prepare the explicit-solvent system and run MD simulations.

The systems were solvated with TIP3P water in boxes shaped as dodecahedrons whose edges are at least 1.2 nm from the protein. 6 Na^+ ions were added to neutralize the charges of the systems.

A.2.2 MD Simulations

All our simulation systems, including WT and two mutants, were minimized with the steepest descent algorithm. The systems were considered as energy minimized when the maximum force becomes 1000 (10.0 kJ/mol). The cutoffs for both long range electrostatic and Van der Waals interactions were set to 1.0 nm. The systems were equilibrated in two phases: 1) NVT equilibration and 2) NPT equilibration. In both phases, we fixed both protein and the ligands with hard position restraints. We equilibrated the system at a temperature of 310K, which was maintained by the velocity-rescaling temperature-coupling algorithm. We increased the cutoffs of non-bonded interactions to 1.2 nm. LINCS constraint algorithm is used and the length of each phase is 1 ns. Particularly in the NPT phase, we adopted Parrinello-Rahman pressure coupling algorithm to maintain the pressure at 1.0 bar.

After the systems were equilibrated, production runs were conducted for 400 ns. A frame was collected every 20 ps. In all the above procedures, periodic boundary conditions and Verlet cutoff-scheme were used.

A.2.3 Data Analysis

The collected production trajectories were analyzed with GROMACS utility tools:

1. Root-mean-square fluctuations of different trajectories were calculated with `gmx rmsf`.

2. Covariance matrices were computed with `gmx covar` under the assumption that each frame can be represented with their CA coordinates.

3. The patterns of fluctuations were analyzed with PCA (See Chapter 2) based on the covariance matrices. PCA was carried out with `gmx anaeig`.

A.3 Preliminary Results[1]

A.3.1 Fluctuations Enhanced by the Mutations

By computing the RMSF for each C-α atom in all trajectories, we found the mutations made in either the ABD or the C2 domain enhanced the fluctuations of some specific part of the protein. As shown in Figure A.2, both mutants have larger fluctuations in the RBD and parts of niSH2 domains.

A.3.2 Covariance between Different Domains

To understand how fluctuations of different domains covariate, we computed a type of covariance matrices for each trajectory. The covariance matrix is defined as the followings: 1) X is a $m \times n$ matrix where m is the number of residues and n is the number of frames. The ith row of X is the difference between the coordinate of the C-α atoms of frame i and the average frame. 2) Then the covariance matrix is defined as $X^T X$. By its defintion, the covariance matrix has a shape of $m \times m$, which represents the covariances of motions of any residue pair. Covariance matrices are plotted in Figure A.3. We found the fluctuations of multiple domains are more correlated in the mutants than in the WT. In particular, strong covariance signals were detected in R38CR88Q between RBD and Helical, and RBD and iSH2. With the mutation N345K involved, more cooperative motions between domains are observed. This result implies that local interactions affected by oncogenic mutations can be propagated to other distant sites of the protein and change their conformational dynamics.

[1]Since this research is not finalized, only a few preliminary results are presented in this section.

A.3.3 Mode of Motions

In order to gain a direct understanding on how the covariated motions occurred, we applied the PCA algorithm to figure out some major patterns of the protein fluctuation. Figure A.4 shows the first two modes of motion of each trajectory. Comparison of the first modes between R38CR88Q and WT and between R38CR88QN345K and WT shows that the amplitudes of motion in the ABD, RBD and the Kinase domain as well as the nSH2 domain are elevated. Since the mutation R38CR88Q lies in the interface between the ABD and the iSH2 domain, it is not surprised to see that the moving correlation between these two domain were enhanced. The moving directions of the ABD domain has a negative correlation with the movement of the nSH2 domain and this negative correlation became more salient in the R38CR88QN345K. It is very likely that this correlated movements are formed because of the rigidness of the iSH2 domain (which is composed of two long helices), and its interaction with the C2 domain. This is because that in the mutants, especially R38CR88QN345K, more and longer arrows pointing at the position of nSH2 are found. Following and linking the arrows of the ABD, iSH2, C2, nSH2 and the Kinase domain yields a direct path of how the effects of the mutations are propogated, which definitely will affect the conformation dynamics of the activation and the catalytic loops and further influence catalysis.

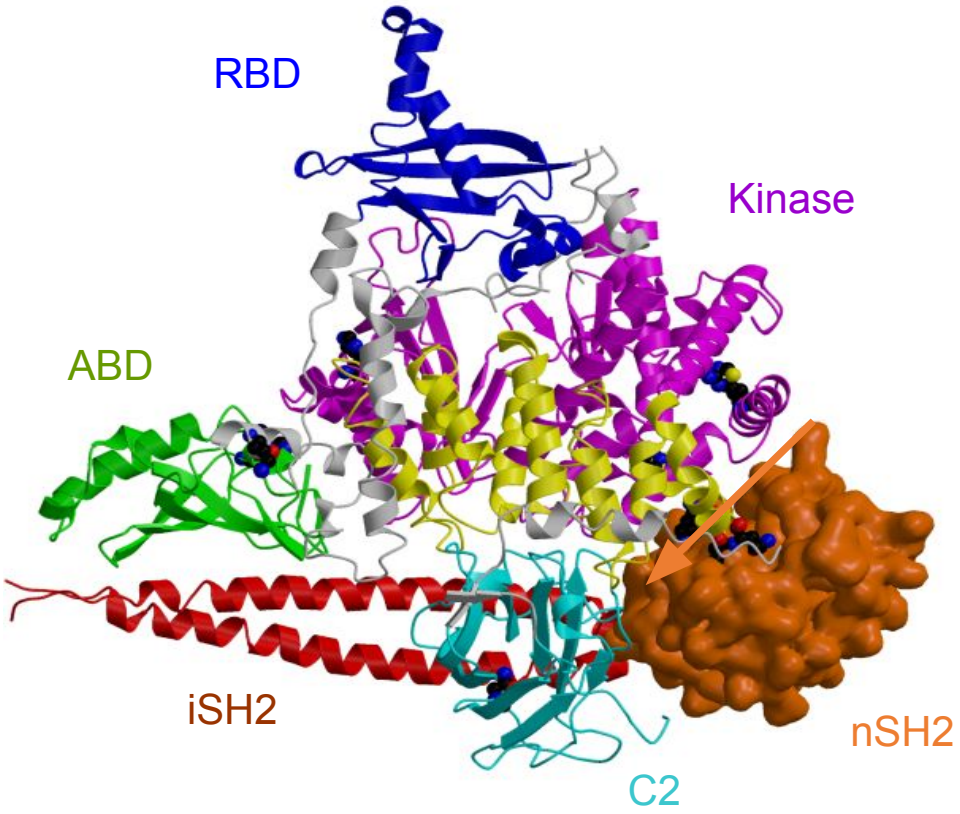

Figure A.1: Structure of the different domains of p110α in complex with niSH2 domains of p85α. This representation is made based on the crystal structure (PDB ID: 3HIZ)

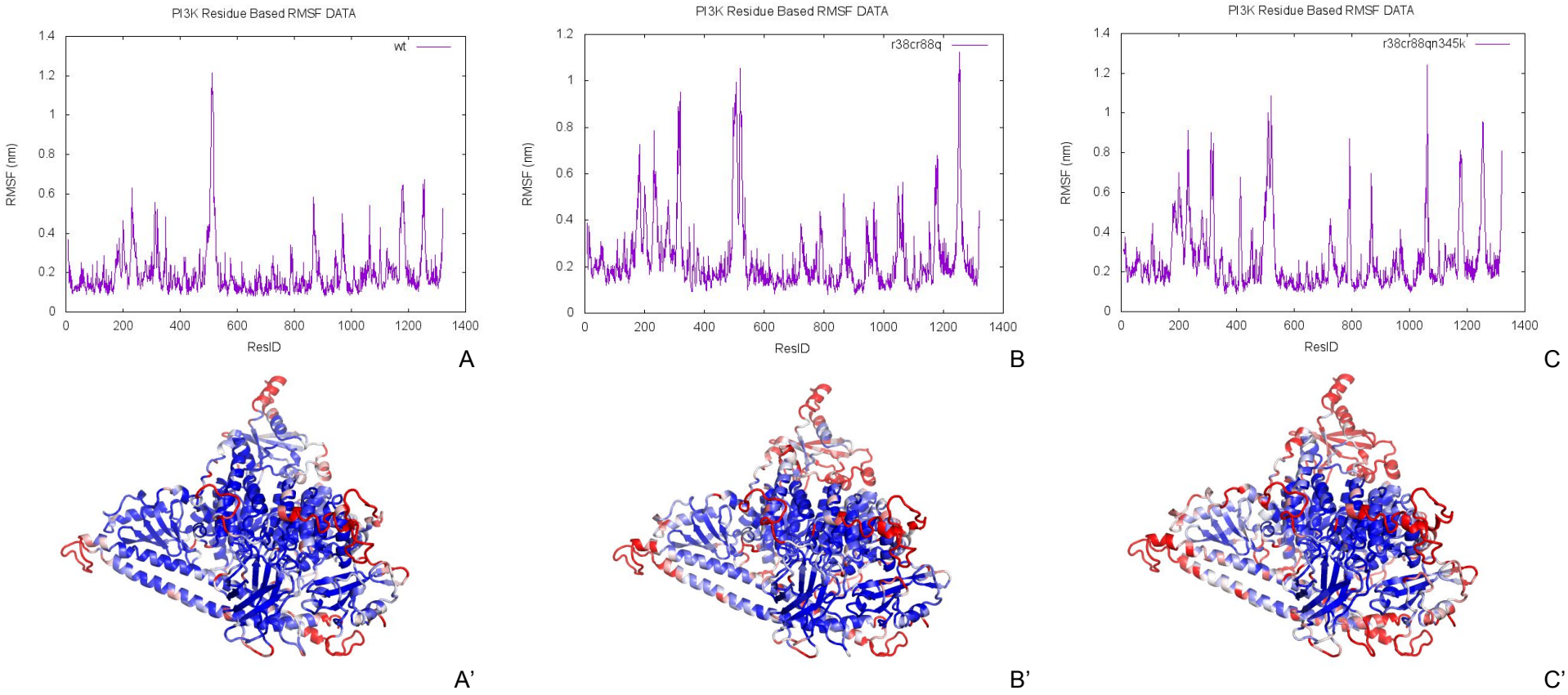

Figure A.2: RMSF and Protein Fluctuations. Three columns of this figure, from left to right, represent the protein fluctuations of WT, R38CR88Q, R38CR88QN345K, respectively. Each column is labeled as a pair, e.g., A and A'. Figure A, B and C give the RMSF per residue, and A', B' and C' reflect the RMSF value on the structure of the protein, which is colored by a color gradient from blue (least fluctuated) to red (most fluctuated).

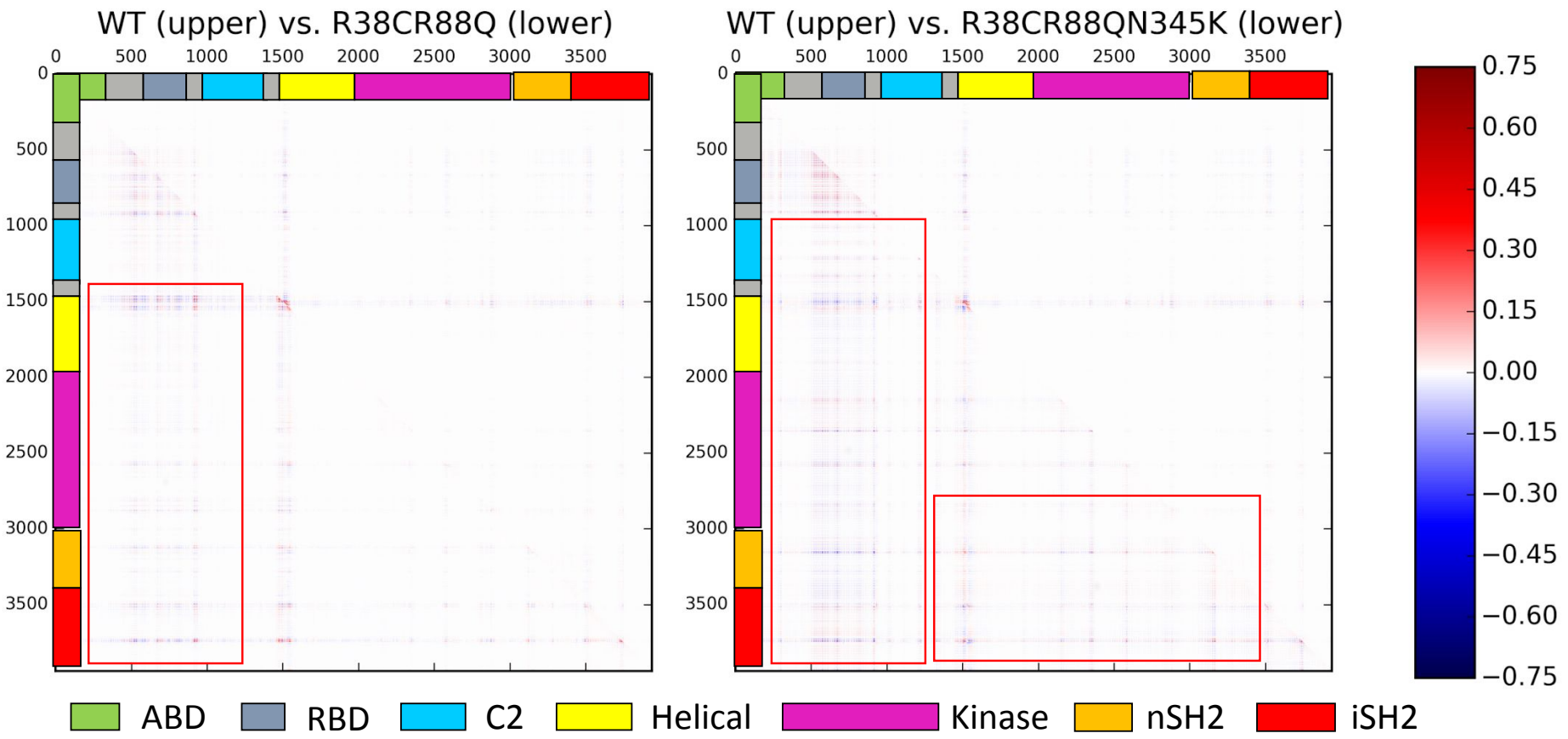

Figure A.3: Covariance Matrices. The covariance matrices of WT, R38CR88Q and R38CR88QN345K are represented by two heatmaps. The elevated signals of the covariance in the mutants are highlighted by the red rectangles.

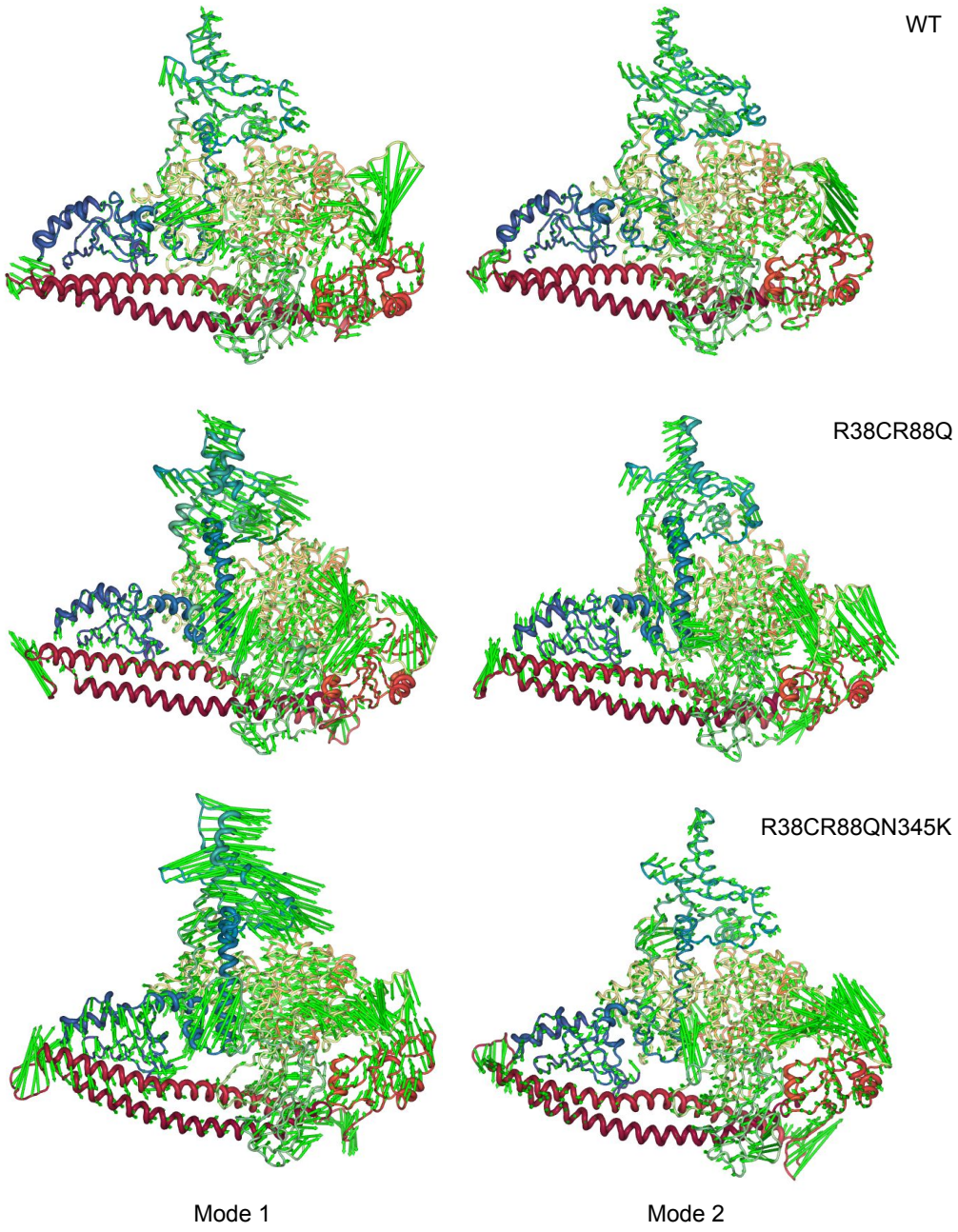

Figure A.4: Visualization of the first two modes of motion idenitified by PCA. Green arrows indicate moving directions of residues in each mode. The length of the arrows represents the amplitude of the motion.

References

[1] Bart VANHAESEBROECK and Dario R ALESSI. "The PI3K–PDK1 connection: more than just a road to PKB". In: *Biochemical Journal* 346.3 (2000), pp. 561–576.

[2] Yardena Samuels, Zhenghe Wang, Alberto Bardelli, Natalie Silliman, Janine Ptak, Steve Szabo, Hai Yan, Adi Gazdar, Steven M Powell, Gregory J Riggins, et al. "High frequency of mutations of the PIK3CA gene in human cancers". In: *Science* 304.5670 (2004), pp. 554–554.

[3] Peter K Vogt, Sohye Kang, Marc-André Elsliger, and Marco Gymnopoulos. "Cancer-specific mutations in phosphatidylinositol 3-kinase". In: *Trends in biochemical sciences* 32.7 (2007), pp. 342–349.

[4] Yun Wang, Åslaug Helland, Ruth Holm, Gunnar B Kristensen, and Anne-Lise Børresen-Dale. "PIK3CA mutations in advanced ovarian carcinomas". In: *Human mutation* 25.3 (2005), pp. 322–322.

[5] Amanda J Philp, Ian G Campbell, Christine Leet, Elizabeth Vincan, Steven P Rockman, Robert H Whitehead, Robert JS Thomas, and Wayne A Phillips. "The phosphatidylinositol 3-kinase p85α gene is an oncogene in human ovarian and colon tumors". In: *Cancer research* 61.20 (2001), pp. 7426–7429.

[6] Hwai Wen Chang, Masahiro Aoki, David Fruman, Kurt R Auger, Alfonso Bellacosa, Philip N Tsichlis, Lewis C Cantley, Thomas M Roberts, and Peter K Vogt. "Transformation of chicken cells by the gene encoding the catalytic subunit of PI 3-kinase". In: *Science* 276.5320 (1997), pp. 1848–1850.

[7] Kurtis E Bachman, Pedram Argani, Yardena Samuels, Natalie Silliman, Janine Ptak, Steve Szabo, Hiroyuki Konishi, Bedri Karakas, Brian G Blair, Clarence Lin, et al. "The PIK3CA gene is mutated with high frequency in human breast cancers". In: *Cancer biology & therapy* 3.8 (2004), pp. 772–775.

[8] Daniel K Broderick, Chunhui Di, Timothy J Parrett, Yardena R Samuels, Jordan M Cummins, Roger E McLendon, Daniel W Fults, Victor E Velculescu, Darell D Bigner, and Hai Yan. "Mutations of PIK3CA in anaplastic oligodendrogliomas, high-grade astrocytomas, and medulloblastomas". In: *Cancer research* 64.15 (2004), pp. 5048–5050.

[9] Ian G Campbell, Sarah E Russell, David YH Choong, Karen G Montgomery, Marianne L Ciavarella, Christine SF Hooi, Briony E Cristiano, Richard B Pearson, and Wayne A Phillips. "Mutation of the PIK3CA gene in ovarian and breast cancer". In: *Cancer research* 64.21 (2004), pp. 7678–7681.

[10] Jong Woo Lee, Young Hwa Soung, Su Young Kim, Hae Woo Lee, Won Sang Park, Suk Woo Nam, Sang Ho Kim, Jung Young Lee, Nam Jin Yoo, and Sug Hyung Lee. "PIK3CA gene is frequently mutated in breast carcinomas and hepatocellular carcinomas". In: *Oncogene* 24.8 (2005), p. 1477.

[11] Douglas A Levine, Faina Bogomolniy, Cindy J Yee, Alex Lash, Richard R Barakat, Patrick I Borgen, and Jeff Boyd. "Frequent mutation of the PIK3CA gene in ovarian and breast cancers". In: *Clinical cancer research* 11.8 (2005), pp. 2875–2878.

[12] Lao H Saal, Karolina Holm, Matthew Maurer, Lorenzo Memeo, Tao Su, Xiaomei Wang, S Yu Jennifer, Per-Olof Malmström, Mahesh Mansukhani, Jens Enoksson, et al. "PIK3CA mutations correlate with hormone receptors, node metastasis, and ERBB2, and are mutually exclusive with PTEN loss in human breast carcinoma". In: *Cancer research* 65.7 (2005), pp. 2554–2559.

[13] ZA Knight and KM Shokat. *Chemically targeting the PI3K family*. 2007.

[14] Chuan-Hsiang Huang, Diana Mandelker, Oleg Schmidt-Kittler, Yardena Samuels, Victor E Velculescu, Kenneth W Kinzler, Bert Vogelstein, Sandra B Gabelli, and L Mario Amzel. "The structure of a human p110α/p85α complex elucidates the effects of oncogenic PI3Kα mutations". In: *Science* 318.5857 (2007), pp. 1744–1748.

[15] Ignacia Echeverria, Yunlong Liu, Sandra B Gabelli, and L Mario Amzel. "Oncogenic mutations weaken the interactions that stabilize the p110α-p85α heterodimer in phosphatidylinositol 3-kinase α". In: *The FEBS journal* 282.18 (2015), pp. 3528–3542.

[16] Diana Mandelker, Sandra B Gabelli, Oleg Schmidt-Kittler, Jiuxiang Zhu, Ian Cheong, Chuan-Hsiang Huang, Kenneth W Kinzler, Bert Vogelstein, and L Mario Amzel. "A frequent kinase domain mutation that changes the interaction between PI3Kα and the membrane". In: *Proceedings of the National Academy of Sciences* 106.40 (2009), pp. 16996–17001.

[17] Michelle S Miller, Oleg Schmidt-Kittler, David M Bolduc, Evan T Brower, Daniele Chaves-Moreira, Marc Allaire, Kenneth W Kinzler, Ian G Jennings, Philip E Thompson, Philip A Cole, et al. "Structural basis of nSH2 regulation and lipid binding in PI3Kα". In: *Oncotarget* 5.14 (2014), p. 5198.

[18] András Fiser and Andrej Šali. "Modeller: generation and refinement of homology-based protein structure models". In: *Methods in enzymology*. Vol. 374. Elsevier, 2003, pp. 461–491.

[19] Xuxiao Zhang, Oscar Vadas, Olga Perisic, Karen E Anderson, Jonathan Clark, Phillip T Hawkins, Len R Stephens, and Roger L Williams. "Structure of lipid kinase p110β/p85β elucidates an unusual SH2-domain-mediated inhibitory mechanism". In: *Molecular cell* 41.5 (2011), pp. 567–578.

[20] Stefano Piana, Kresten Lindorff-Larsen, and David E Shaw. "How robust are protein folding simulations with respect to force field parameterization?" In: *Biophysical journal* 100.9 (2011), pp. L47–L49.

[21] Vincent Zoete, Michel A Cuendet, Aurélien Grosdidier, and Olivier Michielin. "SwissParam: a fast force field generation tool for small organic molecules". In: *Journal of computational chemistry* 32.11 (2011), pp. 2359–2368.

Yunlong Liu

Curriculum Vitae

Education

2012–2018 **Ph.D.**, *The Johns Hopkins University Medical Institution*, Baltimore.
Computational Biophysics

2013–2015 **M.S.**, *The Johns Hopkins University*, Baltimore.
Computer Science

2008–2012 **B.S.**, *Tsinghua University*, Beijing.
Applied Math and Physics

2009–2012 **Bachelor of Economics**, *Tsinghua University*, Beijing.
Economics

Ph.D. thesis

Title *Understanding Protein Mechanism With Computational Approaches*

Advisor L. Mario Amzel

Industrial Experience

Internship

May, 2017 – Aug, 2017 **Software Engineer Intern**, *Google, Inc.*, New York.

May, 2016 – Aug, 2016 **Software Engineer Intern**, *Google, Inc.*, Mountain View.

Award

- 2010 Tsinghua Zheng Geru Scholarship

Publications

[1] Ignacia Echeverria, Yunlong Liu, Sandra B Gabelli, and L Mario Amzel. Oncogenic mutations weaken the interactions that stabilize the p110α-p85α heterodimer in phosphatidylinositol 3-kinase α. *The FEBS journal*, 282(18):3528–3542, 2015.

[2] Giuseppe Ferrandino, Juan Pablo Nicola, Yuly E Sánchez, Ignacia Echeverria, Yunlong Liu, L Mario Amzel, and Nancy Carrasco. Na+ coordination at the na2 site of the na+/i- symporter. *Proceedings of the National Academy of Sciences*, 113(37):E5379–E5388, 2016.

WBSB 601, 725 N Wolfe St – Baltimore, MD, 21205
United States of America
+1 (410) 336 9072 • *yliu120@jhmi.edu*
http://biophysics.med.jhmi.edu/~yliu120/ • *Yunlong Liu*
yliu120